CURSO PRÁTICO 23 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 20,00

INPUT

Vol. 2 Nº 23

NESTE NÚMERO

PROGRAMAÇÃO BASIC

COMO EVITAR ERROS

Importância das mensagens claras e precisas. Rotinas de prevenção de erros. Exclua valores incorretos..... 441

PERIFÉRICOS

COMPUTADORES QUE FALAM

Microcomputadores domésticos com sintetizadores de voz. O que são fonemas. Vantagem de cada tipo. Melhore a pronúncia 446

PROGRAMAÇÃO DE JOGOS

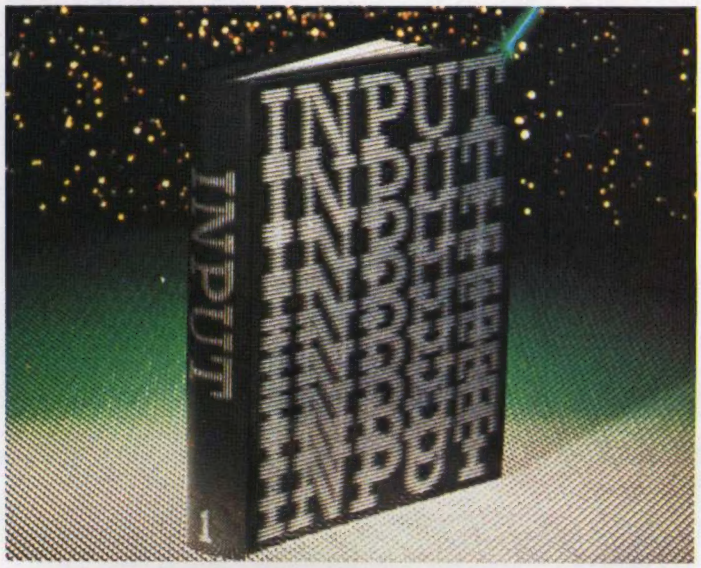
O COMPUTADOR DÁ AS CARTAS

A distribuição das cartas do baralho. Como apostar e quebrar a banca..... 449

PROGRAMAÇÃO BASIC

COMO COMBINAR PROGRAMAS

Não perca tempo digitando rotinas e programas já digitados e testados: utilize alguns comandos e poupe suas energias..... 456



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: **1. Pessoalmente** — por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em São Paulo os endereços são: Rua Brigadeiro Tobias, 773 (Centro); Av. Industrial, 117 (Santo André); e, no Rio de Janeiro: Rua da Passagem, 93 (Botafogo). **2. Por carta** — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidor Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132 (Jardim Tereza) — CEP 06000 — Osasco — São Paulo. **3. Por telex** — Utilize o nº (011) 33670 ABSA. Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações Ltd. — Qta. Pau Varais, Azinhaga de Fetais — 2685, Camarate — Lisboa; Tel. 257-2542 — Apartado 57 — Telex 43 069 JARLIS P.
Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na Agência do Correio. **Atenção:** Após seis meses do encerramento da coleção, os pedidos serão atendidos, dependendo da disponibilidade de estoque. **Obs.:** Quando pedir livros, mencione sempre o título e/ou o autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

Editor
VICTOR CIVITA

REDAÇÃO

Diretora Editorial: Iara Rodrigues

Editor chefe: Paulo de Almeida
Editor de texto: Cláudio A.V. Cavalcanti
Editor de Arte: Eduardo Barreto
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Ailton Oliveira Lopes, Dilvacy M. Santos, José Maria de Oliveira, Grace A. Arruda, Monica Lenardon Corradi
Secretária de Redação/Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz
Secretário Gráfico: Antonio José Filho

COLABORADORES
Consultor Editorial Responsável: Dr. Renato M.E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda. Campinas, SP.
Tradução: Maria Fernanda Sabbatini
Adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Raul Neder Porrelli
Coordenação geral: Rejane Felizatti Sabbatini
Editora de Texto: Ana Lúcia B. de Lucena
Assistente de Arte: Dagmar Bastos Sampaio
COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Ferrucio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atilio Roberto Bonon
Preparador de Texto/Coordenador: Eliel Silveira Cunha

Preparadores de Texto: Ana Maria Dilguerian, Antonio Francelino de Oliveira, Karina Ap. V. Grechi, Levon Yacubian, Maria Teresa Galluzzi, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria do Carmo Leme Monteiro, Maria Luiza Simões, Maria Teresa Martins Lopes.

© Marshall Cavendish Limited, 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986.
Edição organizada pela Editora Nova Cultural Ltda.
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Divisão Gráfica da Editora Abril S.A.

COMO EVITAR ERROS

- USE MENSAGENS CLARAS
- ROTINAS DE PREVENÇÃO DE ERROS
- EXCLUA VALORES INCORRETOS
- INDICAÇÃO DE ERROS

Nem sempre é fácil separar o joio do trigo, quando se trata de programação: programas inteiramente corretos podem apresentar erros inesperados. Aprenda a evitá-los.

A palavra-chave é uso. Não importa se o programa é considerado bom de um ponto de vista estritamente técnico: se ele se mostrar complicado, alguém vai fatalmente acabar tendo dificuldades e cometendo erros.

O segredo para contornar esse problema é fazer programas bem ordenados, de forma que todos os seus módulos sejam claros no que se refere à função e ao modo de uso.

Isso significa oferecer muitas informações e telas explicativas e proteger adequadamente tanto o programa quanto o usuário de erros simples.

O ideal seria colocar no programa proteções contra todos os erros possíveis. Teclas pressionadas equivocadamente, entradas irregulares, valores absurdos — tudo, enfim, que possa atrapalhar a correta execução de um programa deve ser evitado. Para isso, é necessário que se incorporem vários tipos de rotinas de verificação de erros e validação de entradas ao programa. Muitos dos programas mostrados por *INPUT* até o momento fazem uso desses artifícios.

MENSAGENS

Mensagens precisas são muito importantes para ajudar o usuário a compreender de que maneira deve responder quando apresentado, por exemplo, a um menu de opções.

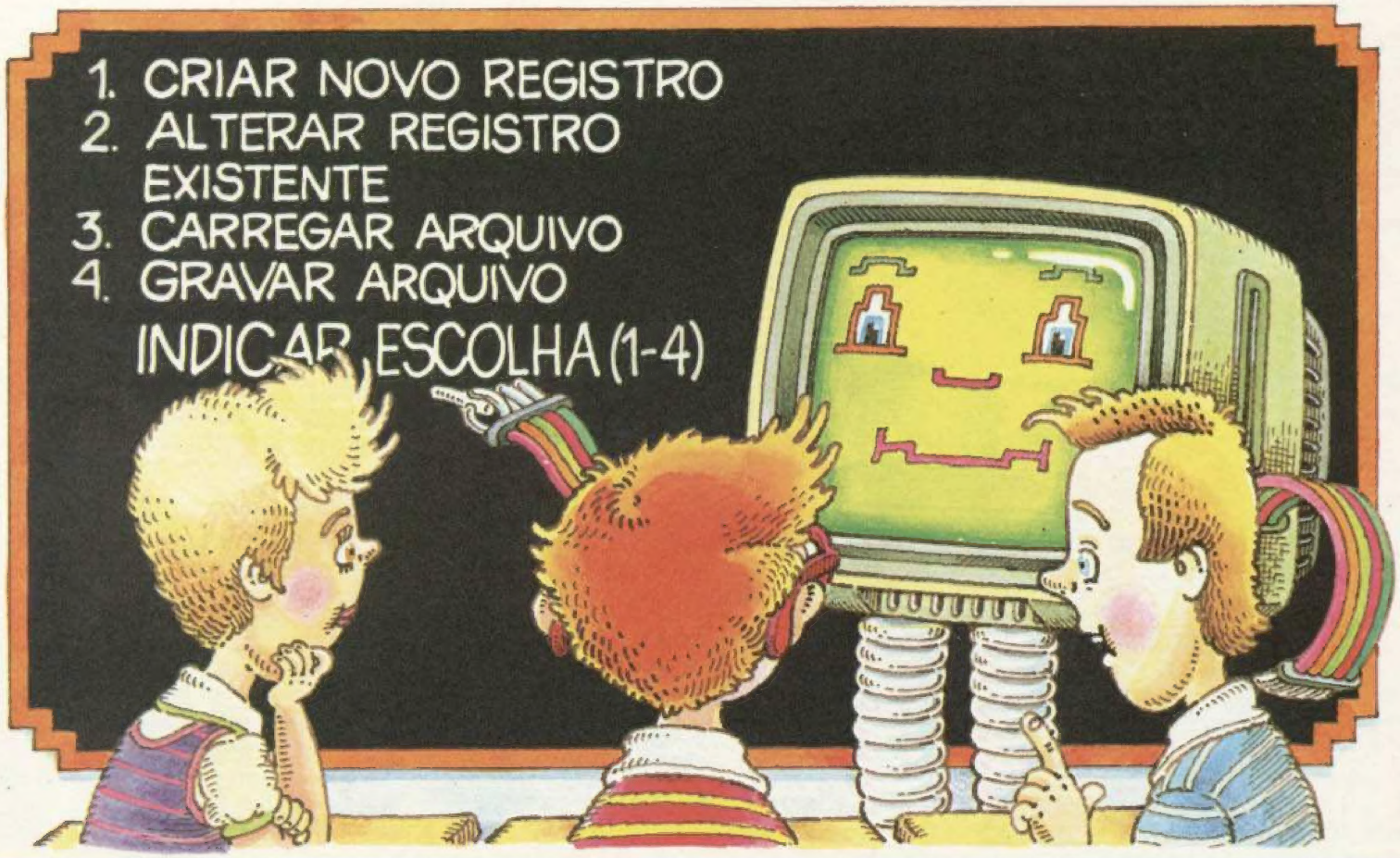
Suponhamos, por exemplo, um menu que mostre as seguintes opções, típicas de um programa que manipula um banco de dados:

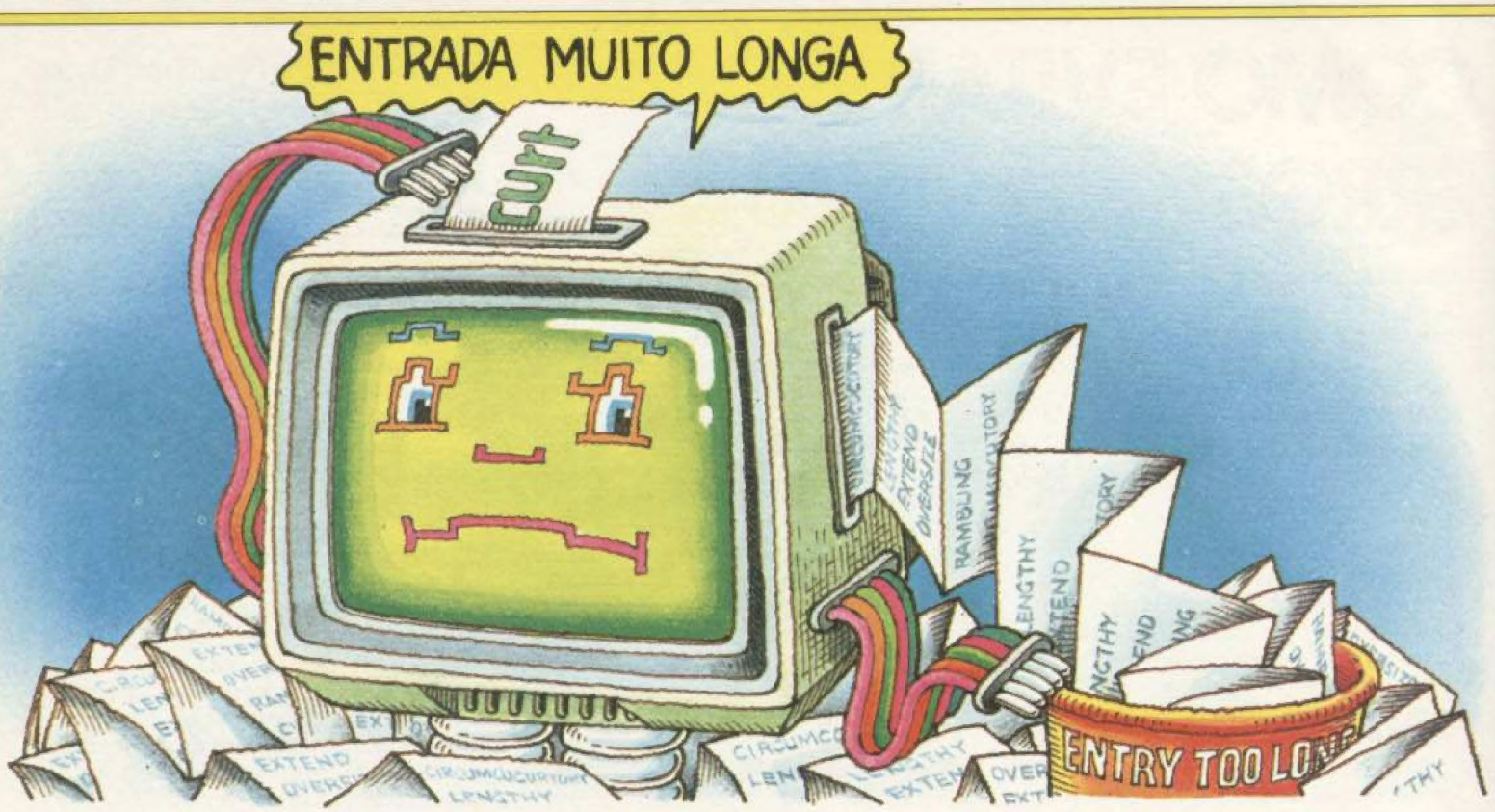
1. Criar um novo registro.
2. Alterar um registro existente.
3. Carregar um arquivo.
4. Gravar um arquivo.

Se for usada uma mensagem do tipo "Selecione uma opção:" ou "Faça sua escolha:", o usuário ficará em dúvida sobre o que fazer. Deve dar entrada ao número da opção ou digitar uma ou mais letras do nome da opção?

Certamente, uma mensagem mais adequada seria "Indique a escolha (1-4)" ou "Tecle 1-4 para fazer sua opção".

Esse tipo de indicação deve ser feito





também em perguntas que exijam uma resposta simples do gênero sim ou não. Assim, em vez de usar alguma coisa do tipo "Tenta novamente?" — onde "sim" pode ser indicado por uma tecla não específica, ou pelo acionamento do botão de disparo do joystick, ou teclando-se S —, especifique as possibilidades: "Pressione o botão de disparo para jogar novamente", ou "Tenta novamente? (S/N)". Estas são formas muito mais diretas e claras de proceder.

Mesmo a mensagem "Pressione qualquer tecla para continuar" pode causar confusão. Vale a pena prepará-la a fim de evitar que alguém queira pressionar algo como **<BREAK>** ou **<STOP>** ou ainda **<ESCAPE>**, correndo, assim, o risco de terminar o programa. Por exemplo, nenhum problema pode acontecer com esta mensagem: "Pressione a barra de espaços para continuar".

Em alguns tipos de computador, a tecla a ser pressionada pode ser evidenciada por meio do caractere invertido na mensagem.

Outra boa opção (que é também uma excelente técnica de programa), não importa a espécie de mensagem, é desativar qualquer tecla que possa atrapalhar ou interromper a execução do programa (isso pode ser feito com o emprego de rotinas que excluam todas as entradas impossíveis).

Para recapitular, ou mesmo para uma simples resposta "sim" ou "não", pode-se usar algo como:

```
90 LET A$=INKEY$
92 IF A$="" THEN  GOTO 90
95 IF A$<>"S" AND A$<>"N"
THEN  GOTO 90
```

```
90 A$=INKEY$
95 IF A$<>"S" AND A$<>"N" THEN
90
```

```
10 GET A$
20 IF A$<>"S" AND A$<>"N"  THEN
GOTO 10
```

Por outro lado, rotinas bem mais sofisticadas podem ser usadas para invalidar a entrada de grupos de valores.

Outra maneira de tornar as coisas mais fáceis para o usuário é restringir a quantidade de dados a serem digitados. Por exemplo: se for suficiente pressionar apenas uma tecla, então prepare o programa de acordo com essa possibilidade. E, para evitar qualquer confusão, utilize sempre o mesmo tipo de mensagem e resposta ao longo de um programa comandado por menus. Assim, se a seleção da opção do menu de abertura for feita pela teclagem de um número, tende a usar o mesmo sistema em todos os outros menus.

Da mesma forma, empregue sempre o mesmo código para cada menu ou lista de opções. Se a terceira opção for "GRAVAR" em um ponto e "SAIR DO PROGRAMA" em outro, alguém pode não ficar muito satisfeito com o seu programa.

Essa regra deve também ser aplicada quando a entrada de dados for necessária (particularmente, quando houver uma longa seqüência). Quando os dados forem restritos a um pequeno grupo e tiverem um padrão simples, pode-se usar uma mensagem múltipla. Um exemplo de dados desse tipo é um arquivo de nomes e endereços, no qual cada nome é seguido de quatro linhas de endereço e um número de telefone.

Por outro lado, é necessário tomar cuidado com coisas mais complexas, como um arquivo de cadastro de clientes, onde há pouca semelhança entre os dados, e o número de entradas muda de registro para registro. Alguns campos podem mesmo ficar vazios.

Uma maneira de limitar o número de erros, neste caso, é fazer uma divisão da entrada de dados, por grupos ou individualmente. Assim, pode-se ter apenas uma mensagem para nome e endereço e outra para cada detalhe específico que vier depois.

Monte as mensagens para que elas apareçam, uma de cada vez, ou em diferentes cores, ou (pelo menos) bem espaçadas umas das outras. Limpar a tela após cada série de entradas é muito

mais eficiente em termos estéticos do que deixar a tela ir rodando para cima (*scroll*) indefinidamente.

Em alguns tipos de programa uma entrada não usual pode chamar uma sub-rotina particular. Assim, em um arquivo de dados, pode ser requerida uma informação adicional para determinados tipos de entrada. Nesse momento, se o usuário, razoavelmente familiarizado com a seqüência normal na qual os dados são entrados, não notar a nova série de entrada de dados, pode cometer diversos tipos de erro.

Nesses casos, são necessárias algumas rotinas de detecção de erros; mas é importante também deixar claro para o usuário que os dados requeridos mudaram. Isso pode ser feito por meio de uma mensagem piscando ou um vídeo inverso, ou mesmo de algum tipo de aviso sonoro, como um bipe, se o seu computador puder emitir sons.

DETECÇÃO DE ERROS

A maneira mais fácil de evitar que o programa incorpore erros é dar ao usuário a possibilidade de aceitar ou rejeitar os dados já digitados. Isso pode ser feito com o auxílio de uma mensagem do tipo: "OS DADOS ESTÃO CORRETOS? (S/N)". Neste caso, se você pressionar a tecla N provocará o reinício da rotina de entrada de dados. Mas isso só será necessário se uma grande quantidade de dados for manipulada.

Se você pretende incorporar a um programa uma rotina de confirmação dos dados, junte as entradas em grupos; desse modo, o programa ficará mais fácil de operar, evitando a repetição da confirmação a cada entrada.

No entanto, apesar de rotinas desse tipo serem eficazes no combate aos erros, os programas têm que ser protegidos contra entradas inadequadas.

Por exemplo, como evitar que letras e números sejam colocados juntos onde somente se requer um desses tipos de informação (ou letra ou número)? Deve-se sempre antecipar possibilidades de erro como esta quando se está montando uma rotina.

LIMITES DE TAMANHO

Na maioria dos programas de controle de arquivos, o tamanho das entradas tem um limite máximo, seja em número de caracteres, seja em valor numérico. Um programa que imprima etiquetas, por exemplo, tem os seus dados restringidos pelo tamanho de cada etiqueta. Afinal, não tem sentido fazer a digitação de um endereço de 25 ou de trinta caracteres se a entrada dispuser de espaço para apenas vinte.

Podemos usar mensagens ou mostrar o limite da entrada com indicadores como caracteres invertidos ou qualquer outro efeito visual que deixe claro qual é o limite máximo aceitável.

Ainda assim, rotinas adicionais devem ser usadas para invalidar entradas que excedam o máximo. Em alguns casos, pode-se truncar o dado no tamanho certo e deixar para a rotina de confirmação a tarefa de verificar se o dado será aceito ou não. Na maioria das vezes, porém, é melhor recomeçar a rotina de entrada no ponto onde houve erro, colocando uma mensagem como "ENTRADA MUITO LONGA! DIGITE NOVAMENTE." ou qualquer coisa parecida.

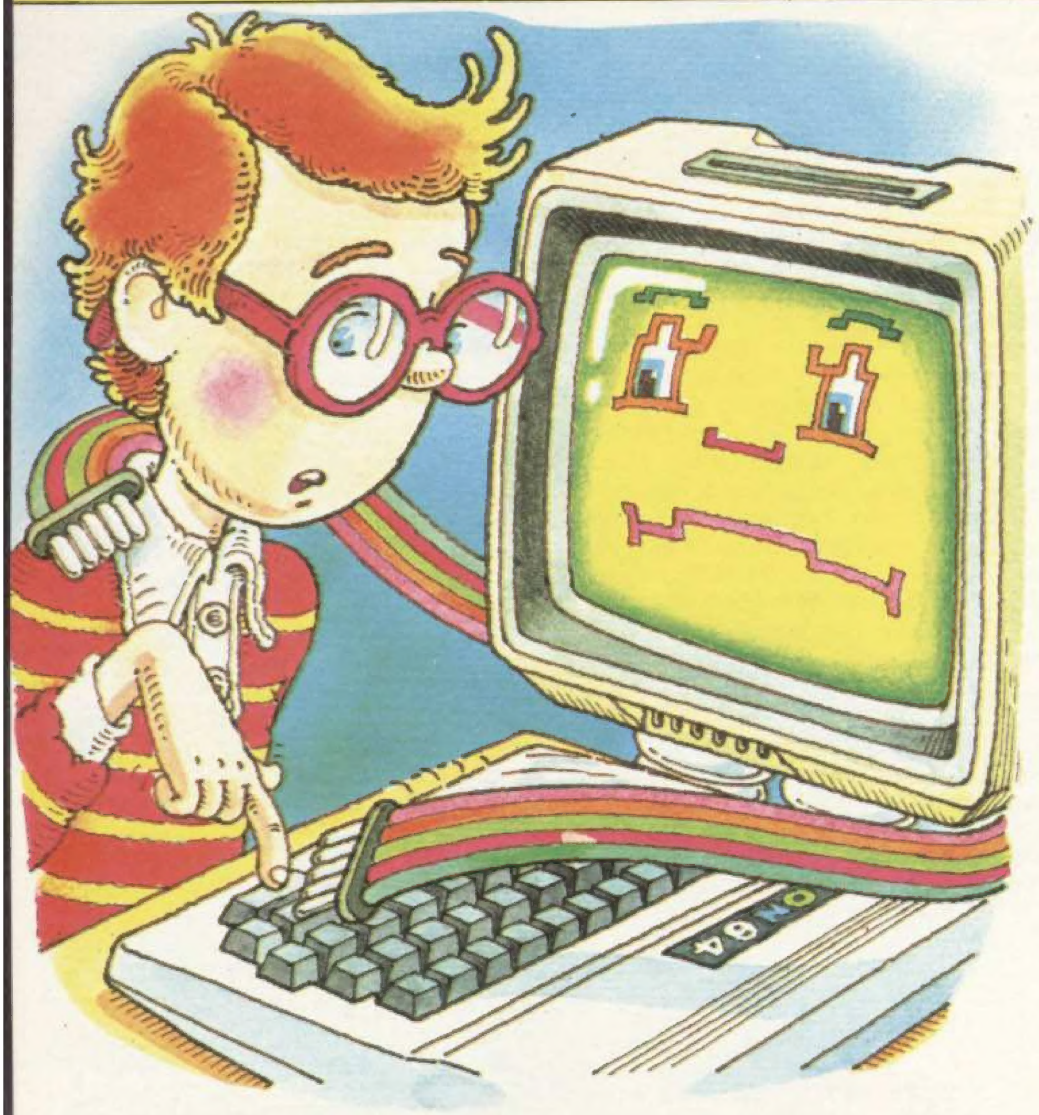
Outra boa razão para se limitar o tamanho das entradas é economizar memória. Afinal, não há sentido, por exemplo, em usar vinte ou mais caracteres para o código de endereçamento postal, quando este nunca ocupa mais de cinco. Procure diminuir o tamanho dos campos tanto quanto possível para economizar memória, especialmente no caso de programas de arquivo.

VALORES INCORRETOS

Estabelecer limites é importante também por outras razões, particularmente em programas que fazem uso de dados numéricos. Suponhamos, por exemplo, que o programa pede três números (ou os lê da memória), e divide a soma dos dois primeiros pelo terceiro. Um pequeno erro de digitação pode colocar um 0 no lugar de um 9 ou fazer com que o valor 0 apareça como resultado de outra operação. Nesse caso, a não ser que o programa esteja protegido contra esse tipo de erro, o resultado será uma mensagem "DIVISÃO POR ZERO" e o término abrupto do programa.

Num caso como o nosso, a proteção exigida é mínima, visto que uma simples rotina de verificação de entradas do teclado é suficiente para restringir entradas inoportunas.

Mas, no caso de cálculos "internos", onde é possível gerar um 0, devemos utilizar rotinas específicas. Este é um caso em que é preciso prever o pior. Assim, convém construir uma rotina com o uso de operadores relacionais (que serão abordados mais adiante, em outra lição). Podemos usar alguma coisa assim:



```
IF A=0 THEN GOTO 10000
```

A linha 10000 será então o início de uma rotina que poderá ajustar o valor para algo diferente de 0 (mas aceitável pelo programa) ou avisar o usuário de que um cálculo impossível está para acontecer. Neste último caso, o programa será redirecionado para a rotina de entrada de dados para que se escolha um valor alternativo.

Uma gama específica de valores pode ser determinada usando-se simplesmente algo como o familiar:

```
IF N>100 OR N<1 THEN...
```

No caso de ser digitado um valor fora da faixa, essa instrução devolverá a execução para o início da rotina de entrada.

Qualquer que seja o seu procedimento dentro do programa, o usuário deve ser avisado sobre qual é a faixa aceitável de dados e, se algum erro ocorrer, ele deve ser informado com uma mensagem adequada.

Desta maneira, ele saberá como proceder. Isso tudo pode ficar em uma sub-rotina acessada só em caso de necessidade.

INDICAÇÃO DE ERROS

Se você der ao usuário instruções claras de como proceder a cada vez que ele for confrontado com uma entrada de dados, uma grande parte dos erros será evitada. No entanto, quando algum erro ocorrer, deve-se informar exatamente a origem do problema para que ele não se repita.

Esse procedimento exige um conjunto de mensagens feitas especialmente para cada situação (não confundir com as mensagens de erro do computador). Estas, por sua vez, podem ser definidas e colocadas em uma sub-rotina especial que só deve ser chamada quando acontecer algum problema.

Essas rotinas são normalmente usadas para indicar valores fora dos limites, duplicação de nomes de um campo-chave de um arquivo de dados, ou ainda entradas incorretas, seja pelo tamanho excessivo, seja pelo uso incorreto de caracteres. Assim, é possível criar mensagens para quaisquer tipos de erros em um programa.

Uma matriz e uma variável indicadora são suficientes para definir um certo número de mensagens e fazer com que o computador escolha a que é adequada para a situação, após identificar o erro ocorrido.

A matriz para tais mensagens deve ser dimensionada logo no início do programa, como parte dos procedimentos de

MICRO DICAS

DETECTE ERROS AUTOMATICAMENTE

Uma das formas de se evitar erros em um programa consiste em equipá-lo com dispositivos de detecção. Esses dispositivos funcionam no sentido de desviar o fluxo do programa, enviando-o a rotinas que corrigem a falha ou previnem o usuário de sua ocorrência.

Para isso, são necessárias instruções especiais. Os micros das linhas TRS-80, TRS-Color, Apple, TK-2000 e MSX, por exemplo, dispõem do poderoso comando de desvio **ON ERROR GOSUB**... Este deve ser colocado em um ponto do programa anterior ao bloco em que se deseja detectar erros de execução. Normalmente, ele é digitado no começo do programa. Sua função consiste em preparar o computador para "saltar" para a linha indicada após o **GOSUB**, sempre que ocorrer um erro. O programador deve então escrever uma rotina (começando na linha indicada) que identifique o erro e adote as medidas pertinentes. Para isso, existem as seguintes instruções suplementares:

- Função **ERR**: retorna o número de código do erro cometido.
- Função **ERL**: retorna o número da linha onde ocorreu a falha.
- Instrução **RESUME**: retorna a rotina de tratamento do erro para um ponto imediatamente após ao da ocorrência desse erro.

Os micros citados acima contam também com o comando **ERROR n**, que provoca uma indicação de erro num ponto do programa (**n** é o número do código). Esse comando testa a instrução **ON ERROR GOSUB** e a rotina de tratamento de erros.

inicialização. Ela deve ser ajustada de tempos em tempos de modo a incluir novas mensagens prevenindo erros que possam vir a ocorrer. Desta maneira, se você resolver usar um conjunto de nove mensagens de erro, este poderia ser um exemplo:

```
10 DIM e$(9,20): FOR z=1 TO 9
: READ e$(z): NEXT z
20 DATA "Dado muito longo!","
Erro de digitacao!"
22 DATA "Senha errada!","Nao
confere!"
24 DATA "Introduza novamente!
","Nao toque!"
26 DATA "Pressione (s)im ou (
n)ao!","Somente numeros!","So
mente letras!"
```

```
10 DIM EM$(9):FOR Z=1 TO 9:READ
 EM$(Z):NEXT Z
20 DATA "DADO MUITO LONGO!","ER
RO DE DIGITACAO!"
22 DATA "SENHA ERRADA!","NAO CO
NFERE!"
24 DATA " REINTRODUZA OS DADOS!
","NAO TOQUE!"
26 DATA "PRESSIONE (S)IM OU (N)
AO!","SOMENTE NUMEROS!","SOMENT
E LETRAS!"
```

Obviamente, você escolherá mensagens adequadas às condições do seu programa. Assim, mais à frente no programa, a cada entrada de dados, pode ser feita a seguinte verificação:

```
1000 LET a$="": LET em=0: INPUT
 a$
1010 IF LEN a$>25 THEN  LET em=
1
1010 IF a$<>"credito" THEN  LET
 em=3
1010 IF a$="5" OR a$="9" THEN
LET em=4
1010 IF a$="" THEN  LET em=5
1010 IF a$=" " THEN  LET em=6
1010 IF a$<>"s" AND a$<>"n" THE
N  LET em=7
1010 IF a$<"0" OR a$>"9" THEN
LET em=8
1010 IF a$<"a" OR a$>"z" THEN
LET em=9
1010 FOR z=1 TO LEN a$: IF a$(z
)="0" THEN  LET em=2
1015 NEXT z
```

```
1000 EM=0:INPUT A$
1010 IF LEN(A$)>25 THEN EM=1
1010 IF A$<>"CREDITO" THEN EM=3
1010 IF A$="5" OR A$="9" THEN E
M=4
1010 IF A$="" THEN EM=5
1010 IF A$=" " THEN EM=6
1010 IF A$<>"S" AND A$<>"N" THE
N EM=7
1010 IF A$<"0" OR A$>"9" THEN E
M=8
1010 IF A$<"A" OR A$>"Z" THEN E
M=9
1010 FOR Z=1 TO LEN(A$):IF MID$
(A$,Z,1)="0" THEN EM=2
1015 NEXT Z
```

Observe que a linha 1010 é opcional, ou seja, você pode escolhê-la ou não, dependendo da situação. Apenas a última opção utiliza uma linha adicional, a 1015. Atenção para a quinta opção, que funciona apenas no TRS-Color e no Spectrum.

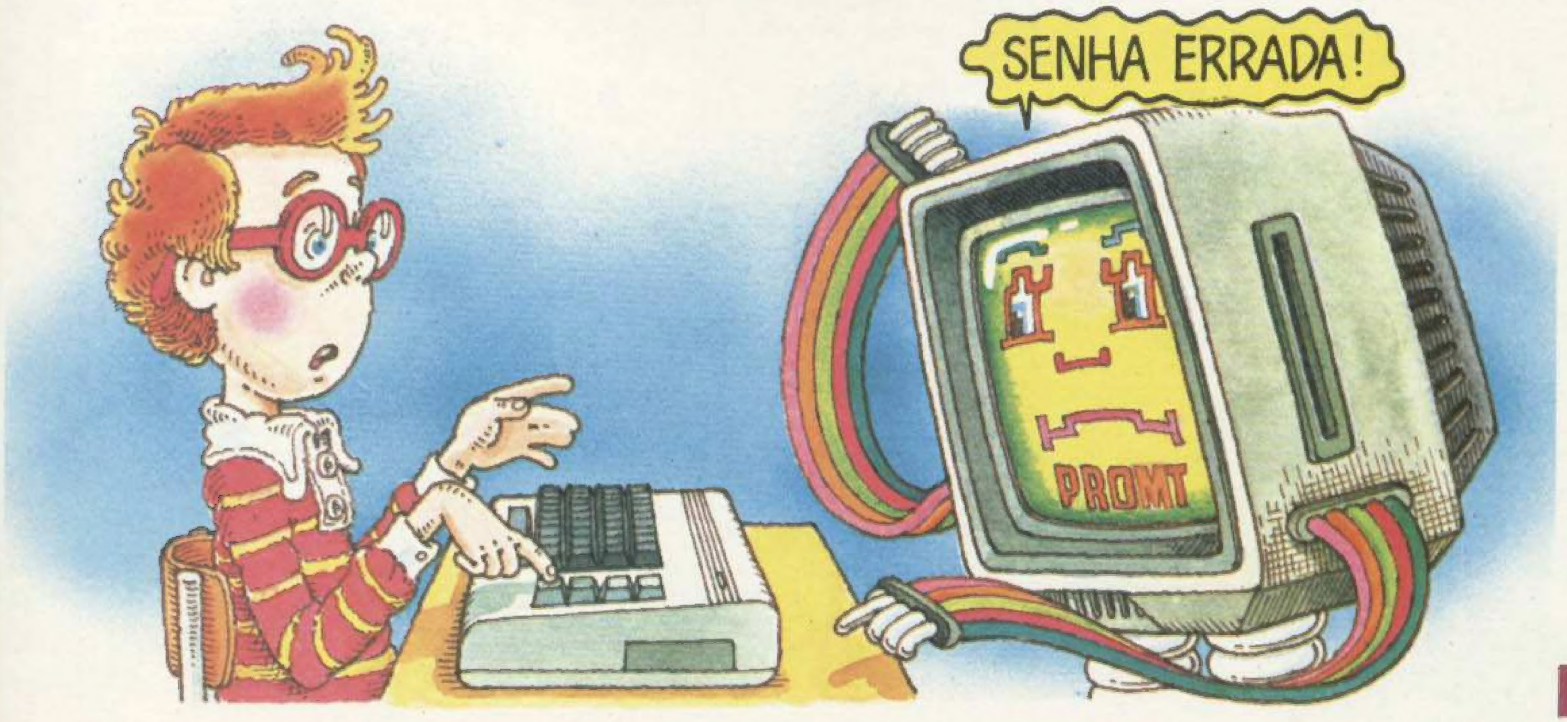
Quando a checagem de entrada revela um erro, a variável **ME** assume um valor particular, relacionado com a mensagem previamente definida que corresponde ao erro.

A primeira opção testa se ocorreu uma entrada maior do que 25 caracteres; a segunda insiste na senha correta. A terceira é típica de uma reconfirmação de informação. A quarta pede que um dado seja redigitado após uma entrada vazia. A opção seguinte responde de modo inesperado quando a barra de espaços é pressionada. A sexta é uma variação da checagem que geralmente é feita após uma resposta sim/não. As duas seguintes verificam se apenas números ou letras foram digitados e a última testa a presença de um O (letra) no lugar de um 0 (número).

O programa prossegue, através de uma sub-rotina que mostra a mensagem adequada ao erro:

```
2000 IF EM>0 THEN  PRINT TAB 6;
e$(em)
```

```
2000 IF EM>0 THEN PRINT EM$(EM)
```



COMPUTADORES QUE FALAM

Já existem sintetizadores de voz para a maioria dos microcomputadores domésticos. Veja aqui a descrição dos tipos mais importantes e os efeitos que você pode obter com cada um deles.

Nos filmes de ficção científica, vemos com freqüência computadores capazes de falar. Você deve se lembrar, entre vários outros, do HAL 9000, o tenebroso e megalomaníaco computador do filme "2001 — Uma Odisséia no Espaço", e os robôs falantes da série "Guerra nas Estrelas". Só muito recentemente, porém, os computadores reais e, particularmente, os microcomputadores pessoais receberam periféricos que lhes conferem o dom da voz. No exterior, já existem sintetizadores de voz para quase todos os tipos de microcomputadores — e a preços bastante baixos, graças à nova geração de chips de circuito integrado, fabricados especialmente para a síntese da voz. Alguns desses sintetizadores podem ser encontrados também no Brasil.

PARA QUE SERVEM?

Por que dar ao micro a capacidade de falar? Bem, antes de mais nada, não há dúvida de que isso seria tremendamente divertido. Basta pensar na infinidade de coisas que você faria com um computador falante: ele poderia contar piadinhas, recitar poemas ou, então, efetuar tarefas mais práticas, como ler em voz alta instruções complexas para um jogo, dispensando-o de abrir o manual a toda hora.

A utilização dos sintetizadores de voz em jogos é, também, muito interessante. Você já deve ter visto alguns videogames e flipers que tagarelam alegremente com o jogador. O sintetizador permite que você aumente a velocidade de interação com o jogo e que economize espaço na tela, se as mensagens ou instruções dirigidas ao jogador forem faladas, e não exibidas. Além disso, como você não precisará ler freqüentes mensagens na tela, poderá concentrar sua atenção nos joysticks e nos gráficos.

É claro que os sintetizadores de voz também têm aplicações mais "sérias", tais como auxiliar uma pessoa cega a utilizar o computador. Um programa especial pode fazer com que o sintetizador pronuncie, letra por letra, ou palavra por palavra, tudo o que é digitado no computador, ou que leia em voz alta listagens de programas ou dados, à medida que aparecem na tela.

Chips de síntese de voz estão começando a ser usados também em automóveis, para lembrar ao motorista que ele esqueceu de colocar o cinto de segurança, avisar-lhe que a pressão do óleo está baixa, e assim por diante. E, sem dúvida, podemos prever que seu uso se expandirá grandemente no ramo dos eletrodomésticos. Já temos uma amostra disso nos fornos de microondas que avisam quando o prato está pronto, ou em secretárias eletrônicas que atendem educadamente aos telefonemas na ausência do morador.

Os sintetizadores de voz também são muito úteis no ensino, tornando divertido aprender a soletrar palavras em qualquer idioma, ou a reconhecer letras e números.

Seu micro pessoal é capaz de tudo isso, desde que você conecte a ele o tipo apropriado de sintetizador de voz.

OS TIPOS PRINCIPAIS

Tecnicamente, existem dois métodos principais para se obter a síntese da voz humana. Eles são bem diferentes entre si, e ambos apresentam vantagens e desvantagens.

O primeiro método utiliza amostras de fala humana: um locutor enuncia letras, números, palavras ou frases inteiras, que são convertidas para números digitais e armazenadas permanentemente em um chip de memória. O segundo método armazena apenas sons elementares (fonemas, sílabas), que um gerador de sons do computador produz, utilizando-os depois para compor palavras e frases mais complexas.

O tipo de sintetizador que armazena amostras de fala humana alcança, naturalmente, uma qualidade de reprodução muito mais fiel. Em compensação, ocupa muito espaço na memória e seu vocabulário é mais restrito.

O chip sintetizador de voz desenvolvido pela Texas Instruments (já disponível no Brasil, com um vocabulário em português) é um bom exemplo deste primeiro tipo. As ondas sonoras (o sinal de voz) são inicialmente convertidas para números digitais por um conversor analógico-digital (ADC), de maneira que possam ser armazenadas em um chip de memória ROM. Posteriormente, quando a palavra ou frase é gerada, o sinal digital armazenado é convertido de volta para o sinal analógico original. Na verdade, não se armazena o sinal de voz original de forma integral, o que exigiria uma quantidade enorme de memória, mas apenas informações sobre o sinal. Consegue-se, assim, uma grande compactação do sinal original. Se a fala fosse integralmente digitada, precisaríamos de cerca de 500 kbytes para 2 minutos de fala — muito mais memória do que a maioria dos micros domésticos dispõe. Felizmente, a técnica de compressão obtém uma redução de 98%, de modo que os dois minutos cabem em um simples chip de ROM.

O segundo tipo de sintetizador de voz é o mais comum. Em vez de armazenar palavras completas, ele gera apenas os fonemas básicos do idioma, como "bá", "ô", "chi" etc. Como estes fonemas também são chamados de alófonos, o sintetizador tornou-se conhecido como sintetizador alofônico.

Existem cerca de sessenta fonemas deste tipo na maioria das línguas ocidentais. Com eles, pode-se construir qualquer palavra ou frase em uma determinada língua. É claro que os fonemas da língua portuguesa são bem diferentes dos da língua inglesa. Assim, em princípio, é possível sintetizar fala em português a partir de um sintetizador fabricado nos Estados Unidos, só que o computador terá um acentuado "sotaque" norte-americano.

Neste método, não se armazena a fala verdadeira, mas apenas a informação digital que é empregada para ativar o gerador de sons interno, o qual produzirá uma aproximação digital do fonema desejado. Cada fonema tem um código, ou endereço inicial de armazenamento, que está disponível em uma tabela de consulta dentro do chip.

Um programa escrito para o sintetizador deve preocupar-se apenas em chamar o código correto, quando ele é necessário. Assim, economiza-se muita memória, pois para cada segundo de fa-

- POR QUE USAR UM SINTETIZADOR DE VOZ
- SINTETIZADOR COM ARMAZENAMENTO DE VOZ
- O TIPO ALOFÔNICO
- O QUE SÃO OS FONEMAS
- VANTAGENS DE CADA TIPO
- CONEXÃO DO SINTETIZADOR
- PROGRAMAÇÃO
- MELHORE A PRONÚNCIA

la são necessários somente 12 bytes de informação digital.

Você já deve ter visto alguns dicionários que utilizam esquemas fonéticos escritos, para indicar a pronúncia das palavras. A representação de cada som muitas vezes corresponde à própria letra ou sílaba, como "te", "cha" etc.; mas, em alguns casos, assume formas bem mais complexas, como "ng", "cz" etc. O número de sons e a forma exata como são indicados dependem, naturalmente, do idioma ou, ainda, do sotaque regional.

VOCABULÁRIO

Os sintetizadores alofônicos têm a vantagem de dispor de um vocabulário praticamente ilimitado, pois qualquer palavra, em princípio, pode ser construída a partir dos fonemas. Mas esses sintetizadores apresentam uma desvantagem: a qualidade da voz, que é baixa, parecendo mais a voz típica de robôs — monótona, sem as modulações de intensidade e freqüência que tornam a fala melodiosa e carregada de diversos tipos de significado (fim de uma frase, exclamação, ironia, ênfase etc.)

A maioria dos sintetizadores que armazenam a fala, por sua vez, dispõe de um vocabulário bastante limitado (algo em torno de cem palavras ou frases curtas), mais a pronúncia das letras do alfabeto, dos números e alguns sufixos e prefixos, como "ésimos". Além disso, ao contrário dos sintetizadores alofônicos, eles não podem ser programados, o que diminui muito sua flexibilidade, ainda que se possa comprar chips adicionais com novos conjuntos de palavras. Mas eles apresentam uma vantagem: como a fala sintetizada é muito mais autêntica, são mais úteis quando mensagens curtas — mas que precisam ser muito claras — devem ser pronunciadas (por exemplo, instruções como "pressione qualquer tecla para começar").

COMO CONECTAR O SINTETIZADOR

Quase todos os sintetizadores de voz são vendidos na forma de cartuchos (semelhantes à expansão de memória dos micros da linha Sinclair; são conectados à interface de expansão na parte de trás do console) ou na forma de placas de circuito impresso, que devem ser encaixadas em um dos soquetes internos do computador (é o caso dos micros da linha Apple). Os sintetizadores que ocupam o único soquete de expansão da máquina geralmente fornecem um soquete de expansão próprio, ao qual se pode ligar um joystick, uma impressora etc.

Outro tipo bastante comum de sintetizador é o que utiliza uma porta serial padrão, do tipo RS-232C, para se comunicar com o computador. Assim, podem ser usados com muitos tipos diferentes de microcomputadores.

Todos esses sintetizadores estarão prontos para operar tão logo sejam ligados ao soquete correto. A saída do som pode ser feita através de uma conexão para um sistema de alta-fidelidade (amplificador e alto-falantes), ou então para a TV. Em ambos os casos, controla-se o volume e a altura do som gerado, o que possibilita torná-lo menos áspero ou "robótico". Outros tipos de sintetizador têm seu próprio sistema de amplificação e de alto-falante, ou utilizam o alto-falante interno do microcomputador. Nesse caso, geralmente é impossível controlar o volume sonoro.

SÍNTESE POR SOFTWARE

Os microcomputadores que possuem geradores sonoros sofisticados oferecem uma alternativa bastante razoável para se obter síntese de voz sem necessidade de hardware especial. Uma máquina bem popular na Europa e nos Estados Unidos, o Commodore 64, tem um poderoso gerador de efeitos sonoros e musicais, que pode ser programado para sintetizar voz por aproximação digital. O sintetizador possibilita o controle de diversos canais de som, simultaneamente, bem como a manipulação de vários parâmetros de modulação de freqüência, envelope e intensidade.

No Brasil, os micros da linha TRS-Color e MSX podem ser usados desta maneira, ou seja, simplesmente adquirindo-se um software específico para síntese de voz. Evidentemente, o tipo utilizado, nesse caso, é o de síntese alofônica, ou por fonemas.

COMO PROGRAMAR O SINTETIZADOR

A forma mais direta de operação da maioria dos sintetizadores consiste, simplesmente, em pronunciar letras e números, cada vez que a tecla correspondente é pressionada no computador. Alguns modelos existentes para o Sinclair e o Spectrum também são capazes de pronunciar por extenso instruções, comandos e funções do BASIC, associados a cada tecla.

Entretanto, para fazer com que o sintetizador emita palavras ou frases completas, é necessário dispor de um software especial — geralmente em linguagem de máquina — que deve ser carregado na máquina antes da utilização do sintetizador. Este software habilita o interpretador BASIC, por meio de uma seqüência de comandos PRINT, a transmitir ao sintetizador a mensagem ou mensagens a serem pronunciadas.

Um sintetizador de voz para os micros da linha Apple, por exemplo, pode ser programado por meio de instruções **PRINT** comuns, dentro de um programa BASIC. Para isso, bastam alguns comandos **POKE** especiais, que habilitam a interface do sintetizador.

Existem três sistemas diferentes para especificar a seqüência de fonemas ou de vocábulos que devem ser pronunciados pelo sintetizador. O mais simples deles utiliza códigos numéricos para identificar os fonemas ou os vocábulos. Estes códigos dependem do chip sintetizador que está sendo empregado; a forma de usá-los dentro de um programa também varia amplamente. Por exemplo, os códigos para "hello" (alô) são 27, 7, 45 e 53. Os números podem ser entrados como uma linha **DATA**:

```
DATA 27,7,45,53
```

... ou como uma cadeia alfanumérica:

```
LET A$="27074553"
```

O manual que acompanha o sintetizador explica detalhadamente, com muitos exemplos, como utilizar os códigos.

A desvantagem desse sistema é que ele é pouco "natural", tornando muito trabalhosa a tarefa de especificar até mesmo seqüências curtas de fonemas ou de palavras. Por isso, desenvolveu-se um segundo tipo de software, mais fácil de usar, que se baseia em algo semelhante aos esquemas fonéticos de pronúncia encontrados nos dicionários. Por este método, as palavras que compõem a frase são escritas com notações especiais para cada fonema. Veja como você poderia dizer "hello" em dois tipos diferentes de sintetizador:

```
10 LET a$ = "he (11)(oo)" :
PAUSE 1
```

```
10 A$="HH1,EH,LL,OW,4"
20 PRINT#A, A$
```

O manual do sintetizador traz uma lista de todos os fonemas, indicando como especificá-los. Como você vê, é bem mais fácil do que usar códigos numéricos mas, em compensação, é preciso conhecer detalhadamente a forma de especificação dos fonemas.

O terceiro sistema é bem mais sofisticado, fazendo a chamada "síntese por transformação direta texto-fala". Por enquanto, programas desse tipo existem somente para alguns idiomas, entre os quais o italiano, o alemão, o japonês, o francês e o inglês. Com eles, basta fornecer um texto escrito de forma correta ao sintetizador, e este produzirá automaticamente a seqüência de fonemas que gerará a voz:

```
10 CALL -435
20 PRINT "HELLO"
```

As regras de conversão texto-fala não são, naturalmente, infalíveis, e a existência de um grande número de exceções, notadamente no inglês, pode produzir os mais absurdos resultados.

Para usar em português um sistema como este, é necessário grafar as frases como um norte-americano as pronunciaria. Por exemplo, como será pronunciado o texto abaixo?

```
20 PRINT "EHDEEHTOHURAH NOHVAH
KOOLTOORAHL"
```

Alguns sistemas de síntese de voz permitem a programação de entonação e ênfase, de modo que a fala saia menos monótona e mais interessante, com pausas e modulações capazes de dar diferentes conotações às palavras — o que é muito importante na linguagem falada. Com isso, pode-se indicar ao computador qual a sílaba tônica de uma palavra, se a frase é interrogativa ou exclamativa etc.

Como se vê, os sintetizadores, além de nos proporcionar momentos de diversão, poderão nos ensinar muita coisa sobre nossa própria linguagem, se forem do tipo alofônico.

O COMPUTADOR DÁ AS CARTAS

- A DISTRIBUIÇÃO DAS CARTAS
- COMO APOSTAR
- PEÇA MAIS CARTAS
- TOTAL DE PONTOS
- TENTE QUEBRAR A BANCA

Num luxuoso cassino, o homem da banca lança um olhar frio e indiferente. Você faz sua aposta e recebe uma nova carta. Há um momento de suspense. Será que você atingiu os 21 pontos?

Além da rotina gráfica apresentada no artigo anterior, precisamos de duas outras para jogar Vinte-e-um: uma que se comunique com o jogador e outra que permita ao computador fazer o papel da banca. A parte que você digitará agora cuida do jogador. Ela faz três coisas: dá as cartas, permite ao jogador fazer apostas e pedir mais cartas, e mostra o total de pontos — soma dos valores das cartas — e de fichas.

A banca solicita ao jogador que aposte, depois de ter dado duas cartas: uma para ele e outra para si mesma. Após receber a segunda carta, o jogador pode "pedir" outra, "comprar" mais uma ou "parar".

O programa tem rotinas para executar as tarefas correspondentes às opções do jogador, distribuindo novas cartas e/ou dobrando a aposta, ou passando a vez de jogar para a banca. Depois de dar uma carta, o programa verifica se o total de pontos ultrapassou 21, ou seja, se o jogador "estourou".

Além disso, o programa prevê duas outras situações: "natural" (quando o jogador faz 21 pontos com apenas duas cartas) e "mão de cinco" (quando cinco cartas reunidas não chegam a somar 22 pontos). Se qualquer delas ocorrer, o jogador será informado e a vez passará para a banca.

S

Adicione estas novas linhas ao programa do artigo anterior. Note que a linha 530 foi apenas aumentada.

```
 90 DIM S(2): LET BET=B
100 DIM O(2): DIM W(2)
500 LET Y=B: LET X=C: LET TF=B
: LET AF=B: LET PF=B: LET FF=B
520 FOR U=C TO 2
530 GOSUB 5500: GOSUB 6000:
 GOSUB 6500
535 IF CC=52 THEN  LET CC=B
540 LET O(U)=C(CC+C)
550 LET X=X+6
560 PRINT  PAPER 7;AT 21,B;"VO
CE TEM ";CP;" FICHAS "
570 IF U=C THEN  INPUT "QUAL E
A SUA APOSTA? ";AM: IF AM>CP
OR AM<C THEN  GOTO 570
590 IF U=C THEN  LET CP=CP-AM:
LET BET=AM
```

```
595 GOSUB 7000: GOSUB 7000:
NEXT U
622 IF S(2)=21 THEN  LET PF=C:
PRINT  PAPER 2;AT 4,18;" NATUR
AL ": GOTO 2500
700 IF (S(C)<16 AND (S(2)<16
OR S(2)>21) OR TF=C OR CP<AM)
THEN  GOTO 705
702 INPUT "COMPRA,PEDE OR SATI
SFEITO? "; LINE D$: IF D$<>"C"
AND D$<>"P" AND D$<>"S" THEN
GOTO 702
703 GOTO 715
705 IF (S(C)>21 OR CP<AM OR TF
=C) THEN  GOTO 710
706 INPUT "COMPRA OU PEDE? ";
LINE D$: IF D$<>"C" AND D$<>"P
" THEN  GOTO 706
708 GOTO 720
710 IF (S(C)<16 AND (S(2)<16
OR S(2)>21)) THEN  GOTO 770
711 INPUT "PEDE OU SATISFEITO?
"; LINE D$: IF D$<>"P" AND D$
<>"S" THEN  GOTO 711
715 IF D$="S" THEN  GOTO 2500
720 IF LEN D$=B THEN  GOTO 700
725 IF D$(C)<>"C" THEN  LET TF
=C: GOTO 905
760 LET BET=BET+AM: LET CP=CP-
AM
770 PRINT  PAPER 7;AT 21,B;"VO
CE TEM ";CP;" FICHAS"
905 LET Z=C(CC)
910 GOSUB 5500: GOSUB 6000:
GOSUB 7000
920 LET X=X+6
921 IF S(C)>21 THEN  GOTO 2000
925 IF X=31 AND S(C)<22 THEN
PRINT  PAPER 2;AT 21,B;" MAO D
E CINCO! ": LET FF=C: GOTO
 2500
930 GOTO 700
2000 IF CP=B THEN  PRINT AT 21,
B;"VOCE PERDEU TODAS AS FICHAS!
": STOP
2010 IF CP>=1000 THEN  PRINT AT
 21,B;"PARABENS! VOCE QUEBROU A
 BANCA!": STOP
2020 PRINT #C;"PRESSIONE 'S' PA
RA OUTRA MAO"
2030 IF INKEY$<>"S" THEN  GOTO
2030
2037 IF PF=C OR DPF=C THEN  GOS
UB 5000
2040 CLS : GOTO 90
2500 STOP
6000 IF VA>10 THEN  LET VA=10
6010 LET S(C)=S(C)+VA: LET S(2)
=S(2)+VA
6020 IF VA=C AND AF=B THEN  LET
 S(2)=S(2)+10: LET AF=C
6030 IF S(C)>21 THEN  PRINT  FL
ASH C; PAPER 7;AT 20,B;" ";TAB
9;"VOCE QUEBROU! ";TAB 31;" ":
LET DPF=B: RETURN
6040 PRINT AT 20,B;" ";TAB 31;"
 "
6050 PRINT  PAPER 7; INK B;AT 2
0,B;"SEU SCORE E' ";S(C);: IF S
(2)<>S(C) AND S(2)<22 THEN  PRI
NT  PAPER 7; INK B;" OU ";S(2)
6060 RETURN
6500 FOR N=10 TO 18
6510 PRINT  PAPER 7; INK 1;AT N
,X;CHR$ 161;CHR$ 161:CHR$ 161;C
HR$ 161;CHR$ 161
6520 NEXT N
6530 RETURN
7000 LET CC=CC+C: IF CC=53 THEN
  LET CC=C
7010 RETURN
```

As linhas 90 e 100 estabelecem a variável **BET** e três matrizes: **S**, que contém os dois totais do jogador (um ás vale 1 ou 11); **O**, que contém as duas primeiras cartas da banca; e **W**, que contém os dois totais correspondentes à soma dessas duas cartas.

A linha 500 estabelece as coordenadas do canto superior esquerdo da carta, e coloca o valor 0 em quatro variáveis indicadoras. **TF** indica se o jogador já pediu cartas. Quando isto acontece pela primeira vez, seu valor passa a ser 1 (segundo as regras, após "pedir" uma carta, o jogador não poderá mais "comprar" outras). **AF** é um sinalizador de ases usado para calcular os dois totais, já que um ás vale 1 ou 11. **PF** é um sinalizador de "naturais", e **FF** é um sinalizador de "mãos de cinco" (as regras do jogo serão apresentadas no próximo artigo).

O laço **FOR...NEXT** entre as linhas 520 e 595 cuida da distribuição das duas primeiras cartas do jogador e da banca. A linha 530 chama duas sub-rotinas: a da linha 6000 atualiza o total à medida que as cartas são dadas, e a da linha 6500 desenha a parte de trás das cartas da banca.

Analisemos essas sub-rotinas mais detalhadamente. A linha 6000 verifica se a carta em questão é uma carta de figura. Se for, seu valor será 10. A linha 6010 soma o valor da carta aos dois totais da matriz **S**. A 6020 verifica se a carta é um ás. A 6030 averigua e informa se o jogador "estourou".

Depois que o programa retorna ao seu curso principal, a linha 535 verifica se a carta atual é de número 52; se isso acontecer, o número da carta será mudado para 0 (nosso baralho tem 52 cartas, numeradas de 1 a 52). A linha 540 coloca as cartas da banca na matriz **0**. A 550 adiciona seis posições à coordenada X da carta seguinte.

O número de fichas que o jogador ainda tem é mostrado pela linha 560. A linha 570 pede para que seja feita a aposta e verifica se há fichas suficientes. O valor da aposta é colocado na variável **BET** e subtraído do total de fichas pela linha 590. As variáveis **AM** e **BET** são necessárias para o caso de o jogador vir a "comprar" cartas.

Os dois **GOSUB 7000** da linha 595 selecionam a carta atual (**CC**), assegurando seu retorno ao início do monte de cartas para que este não acabe. O laço **FOR...NEXT** termina aqui.

A linha 622 verifica se as duas primeiras cartas do jogador fazem um "natural", ou seja, um total de 21 pontos. O sinalizador de "naturais", **PF**, se torna 1 e o programa pára (a linha 2500 será apresentada no próximo artigo).

As linhas 700, 705 e 710 testam as diversas combinações de totais de pontos, o indicador **TW** e o número restante de fichas. Sua função é selecionar as opções para o jogador. Por exemplo, suponhamos que o total de pontos do jogador seja maior que dezesseis e menor que 21. Se ele ainda não tiver pedido nenhuma carta e o número de fichas for

suficiente, a linha 720 colocará à sua disposição as opções de "parar", "pedir" ou "comprar" cartas. Se, ao contrário, ele já tiver "pedido" alguma carta, não poderá mais "comprar" e apenas as opções de "parar" ou "pedir" mais cartas lhe serão dadas pela linha 711. A outra combinação de opções — "comprar" ou "pedir" mais cartas apenas quando nenhum dos totais em **S** ultrapassar dezesseis — é oferecida pela linha 706. Se o jogador resolver parar a essa altura, essa versão inacabada do programa terminará.

Caso apenas **<ENTER>** tenha sido pressionado, sem a letra correspondente à opção, a linha 720 repetirá as opções. A linha 725 acionará o indicador **TW** caso o jogador não tenha "comprado" cartas. Se tiver, o programa irá para a linha 905.

Se o jogador preferir comprar uma carta, a linha 760 modificará o valor da aposta **(BET)** e do restante das fichas **(CP)**. A linha 770 mostra quantas fichas sobraram. As linhas de 905 a 920 mostram as cartas restantes, que foram "compradas" ou "pedidas". A linha 921 atualiza os pontos do jogador. Caso ele tenha "estourado", o programa irá para a linha 2000, onde será feita uma série de verificações.

A linha 925 verifica se o jogador tem uma "mão de cinco", examinando a posição da última carta dada ao jogador: se esta corresponder à quinta carta, e o total for inferior a 22, isso significa que o jogador tem uma "mão de cinco". O indicador **FF** será então acionado e o programa irá para a linha 2500. Se o jogador ainda não tiver parado de "pedir" cartas nem conseguido uma "mão de cinco", a linha 930 fará o programa voltar à linha 700.

QUEBRANDO A BANCA

Se o jogador perder todas as fichas, será informado pela linha 2000 e o jogo acabará. Se ganhar mais de 1000 fichas, ele "quebrará a banca", sendo informado pela linha 2010. Se isso acontecer, o jogo terminará.

Se o jogador tiver menos de 1000 fichas, a linha 2020 perguntará se ele deseja outra "mão". **PRINT 1** é usado para colocar a mensagem na primeira linha da parte inferior do vídeo.

Quando um dos jogadores obtiver um "natural", as cartas serão embaralhadas novamente pela linha 2037, que chamará a sub-rotina adequada. Se ele quiser jogar outra vez a linha 2040 limpará a tela, reiniciando o jogo.

Apague as linhas 190 e 200 do programa anterior e adicione as próximas linhas.

```
84 R=RND(-TIME):MN=100
86 CX=31:CY=1:PL=0:DL=0:NC=2:DA
=0:PA=0:P5=0:TW=0:PF=0:NA=N:BT=
0
210 GOSUB 3000
220 D1=N:GOSUB 1000:FOR I=107 T
O 180 STEP 3:LINE (30,I)-(74,I+
2),6-8*(I/2=INT(I/2)),BF:NEXT
240 GOSUB 5000:GOSUB 6500
250 GOSUB 8000:GOSUB 5000:GOSUB
 7000
260 GOSUB 5000:PRESET(8,80):PRI
NT#1,"Use as setas para apostar
.":PRESET(8,92):PRINT#1,"Depois
 aperte <RETURN>":CLOSE1
265 A$=INKEY$:IF A$<>CHR$(28) A
ND A$<>CHR$(29) AND A$<>CHR$(31
) AND A$<>CHR$(30) AND A$<>CHR$
(13) THEN 265
266 IF ASC(A$)=30 THEN BT=BT+10
:GOSUB 5000:GOSUB 8000:GOTO 265
267 IF ASC(A$)=31 THEN BT=BT+10
*(BT>9):GOSUB 5000:GOSUB 8000:G
OTO 265
268 IF ASC(A$)=28 THEN BT=BT+1:
GOSUB 5000:GOSUB 8000:GOTO 265
269 IF ASC(A$)=29 THEN BT=BT+1*
(BT>0):GOSUB 5000:GOSUB 8000:GO
TO 265
270 GOSUB 5800:BT=INT(BT):IF BT
<1 OR BT>MN THEN 240
280 OB=BT:MN=MN-BT
290 CX=63:GOSUB 3000
300 D2=N:GOSUB 1000:LINE (61,10
6)-(105,185),12,BF:FOR I=107 TO
 180 STEP 3:LINE (62,I)-(105,I+
2),6-8*(I/2=INT(I/2)),BF:NEXT
320 IF PL=11 AND PA=1 AND NC=2
THEN 650
330 CX=CX+32
340 PT=PL+10*(PA AND(PL<12))
350 IF PL>21 THEN 680
360 IF NC=5 THEN GOSUB 5000:GOS
UB 6000:PRINT#1, "VOCE TEM UMA
MAO DE CINCO":P5=1:GOSUB 5500:G
OTO 500
370 GOSUB 5000:GOSUB 6000
380 GOSUB 8000
390 GOSUB 5000:GOSUB 7000
400 IF TW=0 THEN GOSUB5000:PRES
ET(8,80):PRINT#1,"(C)ompra ou":
CLOSE1
410 GOSUB 5000:PRESET (8-12*8*(
TW=0),80):PRINT#1,"(P)ede mais
ou acha":PRESET(8,92):PRINT#1,"
(S)uficiente ?"
430 A$=INKEY$:IF A$<>"P" AND (A
$<>"S" OR PT<16) AND (A$<>"C" O
R TW=1) THEN 430
440 IF A$="S" THEN GOSUB 5800:G
OTO 490
450 IF A$="P" THEN TW=1:GOSUB 5
800:GOTO 480
460 IF MN<OB THEN GOSUB 5800:GO
SUB 5000:PRESET(8,80):PRINT#1,"
Você não tem":PRESET(8,92):PRIN
T#1,"tanto dinheiro assim":GOSU
B 5500:GOTO 400
470 BT=BT+OB:MN=MN-OB:GOSUB 700
0:GOSUB 5000:GOSUB 8000:GOSUB 5
000:GOSUB 5800
480 NC=NC+1:GOSUB 3000:GOTO 320
490 IF PT<16 THEN GOSUB 5000:PR
ESET(8,80):PRINT#1,"Você não po
de parar tendo apenas ";PT:GOSU
B 5500:CX=CX-32:GOTO 370
500 GOTO 750
650 GOSUB 5000:GOSUB 6000:PRESE
T(8,88):PRINT#1," VOCÊ TEM UM N
ATURAL":GOSUB 5500:PF=1:GOTO 86
680 GOSUB 5000:GOSUB 6000:PRESE
T(8,88):PRINT#1,"Você estourou
e perdeu a aposta":GOSUB 5500
750 GOSUB 5000:GOSUB 6000:IF MN
>999 THEN PRINT#1,"PARABENS, VO
CÊ QUEBROU A BANCA":CLOSE1:PLAY
 "T255ADFBFEADC":GOSUB 5500:GOT
O 790
```

```
760 PRESET(8,80):PRINT#1,"Apert
e <RETURN> para":PRESET(8,92):P
RINT#1,"jogar novamente"
770 IF INKEY$<>CHR$(13) THEN 77
0
780 CLOSE1:GOTO 86
2005 LINE (CX-2,CY-1)-(CX-2,CY+
71),12
3000 GOSUB 1000:GOSUB 2000:IF N
M>10 THEN PL=PL+10 ELSE PL=PL+N
M
3010 IF NM=1 THEN PA=1
3020 RETURN
5000 OPEN "GRP:" FOR OUTPUT AS
#1:COLOR15:PRESET(8,88):RETURN
5500 CLOSE1:FOR I=1 TO 2000:NEX
T:LINE (0,80)-(255,106),12,BF:R
ETURN
5800 CLOSE1:LINE (0,80)-(255,10
6),12,BF:RETURN
6000 LINE (204,0)-(255,23),12,B
F:PRESET(204,16):PRINT#1,"PONTO
S":PRESET(204,0):PRINT#1,PL;:IF
 PA=1 AND PL<12 THEN PRESET(204
,8):PRINT#1,"OU";PT
6010 RETURN
6500 LINE (204,0)-(255,23),12,B
F:PRESET(204,16):PRINT#1,"PONTO
S":PRESET(204,0):PRINT#1,PL;:IF
 PA=1 THEN PRESET(204,8):PRINT#
1,"OU 11"
6510 RETURN
7000 LINE (204,32)-(255,55),12,
BF:PRESET(204,48):PRINT#1,"FICH
AS":PRESET(204,32):PRINT#1,MN:C
LOSE#1:RETURN
8000 PRESET(204,132):PRINT#1,"A
POSTA":LINE (204,148)-(255,155)
,12,BF:PRESET(204,148):PRINT#1,
BT:CLOSE#1:RETURN
```

A linha 86 estabelece o valor inicial de uma série de outras variáveis: **CX** e **CY** são as coordenadas do canto superior esquerdo da carta. **PL** e **DL** são os totais do jogador e da banca, respectivamente. **NC** é o número de cartas que um deles tem, conforme aquele que esteja jogando no momento. **DA** e **PA** são indicadores que passam a valer 1, caso a banca ou o jogador tenham um ás. **P5** é outro indicador que passa a valer 1, caso o jogador tenha uma "mão de cinco cartas". **TW** mostra se o jogador já pediu uma carta; caso isso tenha acontecido, as regras impedem que ele "compre" novas cartas. **PF** indica se o jogador tem um "natural" (21 pontos em duas cartas) e **NA** é o número da primeira carta distribuída pela banca.

A sub-rotina que fica entre as linhas 3000 e 3020 é chamada pela linha 210. Uma carta é mostrada na tela; se ela for maior que 10, o total de pontos do jogador será aumentado em dez pontos; caso contrário, o valor real da carta será somado ao total. A linha 3010 verifica se a carta é um ás e modifica o valor de **PA** (indicador de ases do jogador) em caso de necessidade.

A BANCA DÁ AS CARTAS

O programa retorna da sub-rotina 3000 para a linha 220. **D1** é a primeira carta da banca. A sub-rotina 1000 verifica qual o seu valor e naipe. O **FOR...NEXT** desenha o reverso da primeira carta da banca.

Para facilitar a impressão de mensagens na tela de alta resolução, foram feitas várias sub-rotinas no final do programa. A da linha 5000 abre um arquivo "**GRP:**" para que possamos imprimir na tela gráfica por meio de **PRINT 1**; a cor usada é o branco e **PRESET(8,88)** estabelece a posição onde vai ser impressa a mensagem. A sub-rotina da linha 5500 fecha o arquivo aberto pela sub-rotina anterior e apaga a mensagem escrita no meio da tela, usando **LINE,BF**; isso é feito após uma pequena pausa por um laço **FOR...NEXT** para que o jogador tenha tempo de ler a mensagem. A sub-rotina da linha 5800 é igual à da linha 5500, só que não provoca uma pausa.

A sub-rotina da linha 6000 mostra o total de pontos do jogador no canto superior direito da tela. A da linha 6500 exibe, no mesmo local, o valor da primeira carta do jogador. A sub-rotina da linha 7000 mostra o total de fichas que restam ao jogador e a da linha 8000, o valor da aposta feita.

A linha 240 revela os pontos do jogador com o auxílio das sub-rotinas. A linha 250 mostra o valor da aposta — que ainda é zero — e o total de fichas. Estas linhas usam a sub-rotina 5000 para abrir o arquivo que possibilita a impressão na tela gráfica.

A linha 260 solicita ao jogador que faça sua aposta. O arquivo #1 é fechado para evitar problemas — não se deve deixar nunca um arquivo aberto. As linhas 265 a 269 permitem o uso das teclas de controle do cursor para fazer apostas: as setas "direita" e "esquerda" subtraem ou somam 1 ao valor da aposta, as setas "para cima" e "para baixo" somam ou subtraem 10.

A linha 270 apaga a mensagem anterior e também verifica se a aposta foi válida; caso contrário, ela tem que ser feita novamente. Seu valor é subtraído do total de fichas do jogador pela linha 280.

A linha 290 traça a segunda carta do jogador e a linha 300 desenha o reverso da segunda carta da banca. A linha 320 verifica se o jogador tem um "natural" (ou seja, 21 pontos em duas cartas). Neste caso, a linha 650 informa o fato e o jogo recomeça.

A linha 330 cuida da posição da carta seguinte. **PT** é o maior total de pontos que pode ser obtido caso uma das cartas seja um ás. O menor total, ou o único total, se não houver ases, será **PL**. Se este for maior que 21, a linha 680 informará ao jogador que ele "estourou". O programa continua na linha 750, onde são impressas algumas mensagens, conforme a situação.

A linha 360 verifica se o jogador tem uma "mão de cinco". Em caso positivo, o sinalizador **P5** será ativado e o jogador informado; o programa continuará então na linha 500.

A linha 370 mostra os pontos do jogador, a 380 exibe a aposta feita e a 390, o restante das fichas.

AS OPÇÕES DO JOGADOR

As opções de "comprar" ou "pedir" cartas e de "parar" são dadas pelas linhas 400 e 410, enquanto as linhas 430 a 450 cuidam da resposta. Se o jogador tentar "comprar" cartas (o que dobra a aposta) sem ter fichas suficientes, a linha 460 dará a resposta adequada. A linha 470 ajusta e mostra o total de fichas antes que outra carta seja exibida pela linha 480.

Por mais que tente, o jogador não conseguirá "parar" antes de obter pelo menos dezesseis pontos, devido às condições do **IF** da linha 430. A linha 490 é na realidade desnecessária e só entrará em ação caso se mude a condição **PT < 16** da linha 430.

A linha 750 atualiza o total de pontos do jogador e verifica se este tem mais de 999 fichas; neste caso, a banca "quebrará" e o jogador será informado, enquanto uma melodia comemora o feito. As linhas 760 e 780 perguntam então se o jogador quer outra "queda".

Apague a linha 170 e acrescente as próximas linhas ao programa do artigo anterior:

```
150 MN = 100
190  HCOLOR= 1: FOR I = 0 TO 19
1: HPLOT 0,I TO 279,I: NEXT
200 CX = 6:CY = 2:PL = 0:DL = 0
:NC = 2:DA = 0:PA = 0:P5 = 0:TW
 = 0:PF = 0:NA = N
210  GOSUB 3000
220 D1 = N: GOSUB 1000: FOR I =
 100 TO 180: HCOLOR= 3 + 2 * (
INT (I / 2) = I / 2): HPLOT 6,I
 TO 56,I: NEXT
230  FOR K = 1 TO 3500: NEXT
240  HOME : TEXT : PRINT "VOCE
TEM ";PL;: IF PA = 1 THEN  PRIN
T " OU 11": GOTO 250
245  PRINT
250  PRINT : PRINT "VOCE TEM ";
MN;" FICHAS"
260  PRINT : INPUT "QUANTO VOCE
 APOSTA ?";BT
270 BT =  INT (BT): IF BT < 1 O
R BT > MN THEN 240
280 OB = BT:MN = MN - BT
290 CX = 60: GOSUB 3000
300 D2 = N: GOSUB 1000: FOR I =
 100 TO 180: HCOLOR= 3 + 2 * (
INT (I / 2) = I / 2): HPLOT 60,
I TO 110,I: NEXT
310  FOR K = 1 TO 4000: NEXT
320  IF PL = 11 AND PA = 1 AND
NC = 2 THEN 650
330 CX = CX + 54
340 PT = PL + 10 * (PA AND (PL
< 12))
350  IF PL > 21 THEN 680
360  HOME : TEXT : IF NC = 5 TH
EN  PRINT "VOCE TEM UMA MAO DE
CINCO":P5 = 1: FOR K = 1 TO 250
0: NEXT : GOTO 500
370  PRINT "VOCE TEM ";PL;: IF
PA = 1 AND PL < 12 THEN  PRINT
" OU ";PT: GOTO 380
375  PRINT
380  PRINT : PRINT "VOCE APOSTO
U ";BT
390  PRINT : PRINT "VOCE TEM ";
MN;" FICHAS"
400  PRINT : IF TW = 0 THEN  PR
INT "(C)OMPRA OU "
410  PRINT : PRINT "(P)EDE MAIS
 ";: IF PT > 15 THEN  PRINT "OU
 ACHA (S)UFICIENTE";
420  PRINT " ?"
430  GET A$: IF A$ <  > "P" AND
 (A$ <  > "S" OR PT < 16) AND (
A$ <  > "C" OR TW = 1) THEN 430
440  IF A$ = "S" THEN 490
450  IF A$ = "P" THEN TW = 1: G
OTO 480
460  IF MN < OB THEN  PRINT : P
RINT "VOCE NAO TEM TANTO DINHEI
RO ASSIM": GOTO 400
470 BT = BT + OB:MN = MN - OB
480 NC = NC + 1: GOSUB 3000: GO
TO 310
490  IF PT < 16 THEN  PRINT : P
RINT "VOCE NAO PODE PARAR TENDO
 APENAS ";PT:CX = CX - 50: GOTO
 370
500  GOTO 750
650  HOME : TEXT : PRINT "VOCE
TEM UM NATURAL": FOR K = 1 TO 1
500: NEXT :PF = 1: GOTO 190
680  HOME : TEXT : PRINT "VOCE
ESTOUROU E PERDEU A APOSTA"
750  PRINT : PRINT "VOCE TEM ";
MN;" FICHAS": PRINT : IF MN > 9
99 THEN  HOME : INVERSE : HTAB
8: VTAB 11: PRINT "VOCE QUEBROU
 A BANCA": FOR I = 1 TO 2000:XX
 =  PEEK ( - 16336): NEXT : GOT
O 790
760  PRINT : PRINT "APERTE <RET
URN> PARA OUTRA MAO"
770  GET A$: IF A$ <  >  CHR$ (
13) THEN 770
780  GOTO 190
3000  POKE  - 16299,0: POKE  -
```

```
16304,0: GOSUB 1000: GOSUB 2000
: IF NM > 10 THEN PL = PL + 10:
 GOTO 3010
3005 PL = PL + NM
3010  IF NM = 1 THEN PA = 1
3020  RETURN
```

Os leitores que possuem vídeo monocromático e que no último artigo preferiram apagar a linha 170 devem substituir a linha 190 por:

```
190 HGR2
```

A linha 150 faz com que o número de fichas (**MN**) do jogador seja 100. A linha 190 limpa a tela.

A linha 200 estabelece os valores iniciais de uma série de variáveis. CX e CY são as coordenadas do canto superior esquerdo da carta. **PL** a **DL** são os totais de pontos do jogador e da banca, respectivamente. **NC** é o número de cartas do jogador ou da banca, conforme aquele que estiver jogando. **DA** e **PA** são indicadores, respectivamente, de que o jogador ou a banca possuem um ás. **P5** é um indicador de que o jogador possui uma "mão de cinco cartas". **TW** revela se o jogador já "pediu" cartas, sendo então utilizado para evitar que ele "compre" cartas, segundo as regras do jogo. **PF** é um indicador de "naturais" (ou seja, pontos em duas cartas). **NA** é o número da primeira carta dada pela banca. Um indicador é uma variável que vale 0 ou 1, de acordo com a situação.

A sub-rotina da linha 3000 é chamada pela linha 210. Uma carta é colocada na tela; se for maior que 10, o total de pontos do jogador será aumentado em 10; caso contrário, o valor da carta será somado aos pontos do jogador. A linha 3010 verifica se a carta é um ás, se necessário.

A BANCA DÁ AS CARTAS

O programa retorna da sub-rotina para a linha 220. Ali, **D1** é a primeira carta da banca e a sub-rotina 1000 é usada para descobrir seu valor e naipe. O **FOR...NEXT** desenha o reverso da carta da banca.

A linha 230 faz uma pausa antes que a linha 240 limpe e ative a tela de textos e informe o total ao jogador. A mensagem adicional "OU 11" é usada somente quando a carta é um ás. O número de fichas é escrito pela linha 250. A linha 260 cuida da aposta do jogador e a 270 providencia para que a aposta seja maior que zero e esteja dentro das posses do jogador (caso contrário, ela terá que ser feita novamente). O valor da aposta é subtraído das fichas do jogador pela linha 280.

A seguir, a linha 290 chama a sub-rotina que distribui as cartas, e a linha 300 desenha a parte posterior da segunda carta da banca.

A linha 320 verifica se o jogador tem um "natural" depois da pausa causada pela linha 310. Em caso positivo, a linha 650 informará o fato ao jogador e o programa recomeçará.

A linha 330 cuida da posição da carta seguinte. **PT** é o maior dos totais de pontos, para o caso de o jogador possuir um ás. Se não houver ases, **PL** será o menor (ou o único) total. Se **PL** for maior que 21, a linha 680 informará ao jogador que ele "estourou". O programa continuará, então, imprimindo as mensagens da linha 750 em diante.

A linha 360 verifica se o jogador tem uma "mão de cinco cartas", colocando 1 em **P5** e criando uma pausa, caso isto seja necessário.

A linha 370 mostra o total contido nas cartas do jogador (dois valores, se houver um ás e o valor menor for inferior a 12). A aposta do jogador é impressa pela linha 380 e a quantidade de fichas pela linha 390.

AS OPÇÕES DO JOGADOR

As opções de "parar", "comprar" ou "pedir" cartas são oferecidas pelas linhas 400 a 420. As linhas 430 a 450 cuidam das respostas do jogador. Se este tentar "comprar" cartas sem ter dinheiro para dobrar a aposta, a linha 460 tratará de informá-lo. A linha 470 acerta o valor da aposta e do total de fichas antes da linha 480 dar a próxima carta.

A condição **PT < 16** da linha 430 impede que o jogador "pare" sem ter pelo menos dezesseis pontos. A linha 490 é desnecessária e só entrará em ação se essa condição for modificada. O programa continua na linha 750, que avisa ao jogador o número de fichas restantes, verificando se a banca foi "quebrada". Se isso acontecer, um ruído será produzido.

As linhas 760 a 780 oferecem ao jogador a opção de jogar outra "mão".

Para que o programa funcione no TK-2000 são necessárias algumas modificações. Em primeiro lugar, daqui para a frente use apenas a **MP** (segunda página de vídeo). Quando o programa estiver completo, qualquer retorno a **MA** (primeira página de vídeo) vai danificá-lo. Então, primeiro digite **MP** e depois <**RETURN**>; carregue o programa inicial e faça as mesmas modificações sugeridas para o Apple com exceção das seguintes linhas:

```
160  HGR2
240  GOSUB 5000: PRINT "VOCE TE
M ";PL;: IF PA = 1 THEN  PRINT
" OU 11": GOSUB 6000: GOTO 250
245  PRINT : GOSUB 6000
250  GOSUB 5000: PRINT "VOCE TE
M ";MN;" FICHAS": GOSUB 6000
260  GOSUB 5000: INPUT "QUANTO
VOCE APOSTA ?";BT: GOSUB 6000
360  GOSUB 5000: IF NC = 5 THEN
  PRINT "VOCE TEM UMA MAO DE CI
NCO":P5 = 1: GOSUB 6000 GOTO 50
0
370  PRINT "VOCE TEM ";PL;: IF
PA = 1 AND PL < 12 THEN  PRINT
" OU ";PT: GOSUB 6000: GOTO 380
375  PRINT : GOSUB 6000
380  GOSUB 5000: PRINT "VOCE AP
OSTOU ";BT: GOSUB 6000
390  GOSUB 5000: PRINT "VOCE TE
M ";MN;" FICHAS": GOSUB 6000
400  GOSUB 5000: IF TW = 0 THEN
  PRINT "(C)OMPRA OU "
460  IF MN < OB THEN  GOSUB 500
0: PRINT "VOCE NAO TEM TANTO DI
NHEIRO ASSIM": GOSUB 6000: GOTO
 400
490  IF PT < 16 THEN  GOSUB 500
0: PRINT "VOCE NAO PODE PARAR T
ENDO APENAS ";PT:CX = CX - 50:
GOSUB 6000: GOTO 370
650  GOSUB 5000: PRINT "VOCE TE
M UM NATURAL": GOSUB 6000:PF =
1
660  GOSUB 4000: GOSUB 5000: IF
 DL = 11 THEN  PRINT "MAS A BAN
CA TEM A MESMA COISA": GOSUB 60
00: GOTO 610
680  GOSUB 5000: PRINT "VOCE ES
TOUROU E PERDEU A APOSTA": GOSU
B 6000
690  GOSUB 5000: IF MN < 1 THEN
  PRINT "VOCE PERDEU TODAS AS F
ICHAS": GOSUB 6000: GOTO 790
700  IF PF = 1 THEN  PRINT "EMB
ARALHANDO AS CARTAS": GOSUB 150
0: GOSUB 6000: GOTO 750
750  GOSUB 5000: PRINT "VOCE TE
M ";MN;" FICHAS": GOSUB 6000: I
F MN > 999 THEN  GOSUB 5000: PR
INT "VOCE QUEBROU A BANCA": FOR
 I = 1 TO 2000: NEXT : GOTO 790
760  GOSUB 5000: PRINT "APERTE
<RETURN> PARA OUTRA MAO": GOSUB
 6000
5000  HCOLOR= 1: FOR I = 84 TO
108: HPLOT 0,I TO 255,I: NEXT :
 HTAB 1: VTAB 12: RETURN
6000  FOR I = 1 TO 1000: NEXT :
 RETURN
```

Note que você terá que procurar na listagem das modificações para o Apple algumas linhas que não estão aqui por falta de espaço. A diferença entre o programa do Apple e do TK-2000 se deve ao fato de este último não possuir uma página só para textos. Assim, as men-

sagens terão que ser impressas na tela gráfica.

A sub-rotina da linha 5000 apaga qualquer coisa que tenha sido escrita antes e posiciona o cursor para a mensagem seguinte. A sub-rotina 6000 provoca uma pausa para que a mensagem possa ser lida. No restante, o programa é igual ao do Apple e as explicações são as mesmas.

As linhas 220 e 300 são um pouco diferentes das do Apple. Elas colocam as cartas da banca mais abaixo, abrindo um espaço para que as mensagens sejam impressas na tela.

T

Acrescente as próximas linhas àquelas que digitou da outra vez:

```
170 MN=100
190 PCLS 6
200 CX=6:CY=11:PL=0:DL=0:NC=2:D
A=0:PA=0:P5=0:TW=0:PF=0:NA=N
210 GOSUB 3000
220 POKE 178,156:D1=N:GOSUB 100
0:LINE (6,108)-(50,180),PSET,BF
230 FOR K=1 TO 2000:NEXT
240 CLS:PRINT" VOCE TEM";PL;:IF
 PA=1 THEN PRINT "OU 11" ELSE P
RINT
250 PRINT:PRINT" VOCE TEM";MN;"
FICHAS"
260 INPUT" FACA SUA APOSTA ";BT
270 BT=INT(BT):IF BT<1 OR BT>MN
 THEN 240
280 OB=BT:MN=MN-BT
290 CX=56:GOSUB 3000:POKE 178,1
56
300 D2=N:GOSUB 1000:LINE(56,108
)-(100,180),PSET,BF
310 FOR K=1 TO 2500:NEXT
320 IF PL=11 AND PA=1 AND NC=2
THEN 650
330 CX=CX+50
340 PT=PL+10*(PA AND(PL<12))
350 IF PL>21 THEN 680
360 CLS:IF NC=5 THEN PRINT" VOC
E TEM UMA MAO DE CINCO":P5=1:FO
R K=1 TO 1500:NEXT:GOTO 500
370 PRINT" VOCE TEM";PL;:IF PA=
1 AND PL<12 THEN PRINT "OU";PT
ELSE PRINT
380 PRINT:PRINT" VOCE APOSTOU";
BT
390 PRINT:PRINT" VOCE TEM";MN;"
FICHAS"
400 PRINT:IF TW=0 THEN PRINT" (
C)OMPRA OU";
410 PRINT" (P)EDE MAIS CARTAS";
:IF PT>15 THEN PRINT" OU (S)ATI
SFEITO";
420 PRINT " ?"
430 A$=INKEY$:IF A$<>"P" AND (A
$<>"S" OR PT<16) AND (A$<>"C" O
R TW=1) THEN 430
440 IF A$="S" THEN 490
450 IF A$="P" THEN TW=1:GOTO 48
0
460 IF MN<OB THEN PRINT " VOCE
NAO TEM TANTO DINHEIRO":GOTO 40
0
470 BT=BT+OB:MN=MN-OB
480 NC=NC+1:GOSUB 3000:GOTO 310
490 IF PT<16 THEN PRINT " VOCE
NAO PODE PARAR AGORA";PT:CX=CX-
50:GOTO 370
500 GOTO 750
650 CLS:PRINT" VOCE TEM UM NATU
RAL":FOR K=1 TO 1000:NEXT:PF=1:
GOTO 190
680 CLS:PRINT" VOCE ESTOUROU E
PERDEU A APOSTA"
750 PRINT:PRINT" VOCE TEM";MN;"
FICHAS":PRINT:IF MN>999 THEN PR
INT "MUITO BEM, VOCE QUEBROU A
BANCA!":SCREEN 0,1:PLAY"T6ADFBF
EADC":GOTO 790
760 PRINT:PRINT" PRESSIONE (S)
PARA RECOMECAR"
770 IF INKEY$<>"S" THEN 770
780 GOTO 190
3000 SCREEN 1,1:GOSUB 1000:GOSU
B 2000:IF NM>10 THEN PL=PL+10 E
LSE PL=PL+NM
3010 IF NM=1 THEN PA=1
3020 RETURN
```

A linha 170 faz com que o número de fichas do jogador seja igual a cem. A linha 190 limpa e dá cor à tela.

A linha 200 estabelece o valor inicial de uma série de variáveis, CX e CY são as coordenadas do canto superior esquerdo da carta. **PL** e **DL** são, respectivamente, os totais de pontos do jogador e da banca. **NC** é o número de cartas de quem estiver jogando no momento, jogador ou banca. **DA** e **PA** são indicadores que passam a valer 1 se a banca ou o jogador tiverem um ás. **P5** é um indicador que sofrerá a mesma modificação caso o jogador consiga uma "mão de cinco cartas". **TW** é um indicador que passará a valer 1 quando o jogador "pedir" uma carta. Segundo as regras, a partir de então ele não poderá mais "comprar". **PF** é um indicador de "naturais" e **NA** o número da primeira carta dada pela banca.

A sub-rotina da linha 3000 é chamada pela linha 210. Uma carta é colocada no vídeo. Se for maior que 10, o total do jogador será aumentado em dez; caso contrário, o valor da carta será somado aos pontos do jogador. Se a carta for um ás, a linha 3010 colocará 1 no sinalizador de ases **PA**.

A BANCA DÁ AS CARTAS

Quando o programa retornar, a linha 220 usará **POKE** para obter o padrão usado no reverso das cartas. **D1** é a primeira carta da banca; a sub-rotina descobre seu valor e naipe. **LINE** desenha a carta e **POKE** cria o padrão.

A linha 230 provoca uma pausa antes que a linha 240 limpe a tela e escreva o total do jogador. O comando **PRINT** muda automaticamente a tela gráfica para a tela de textos. Se a carta for um ás, aparecerá uma mensagem adicional "OU 11". O número de fichas é informado pela linha 250.

A linha 260 cuida da aposta; a linha 270 verifica se esta é válida; caso contrário, ela deve ser refeita. A aposta é subtraída das fichas do jogador pela linha 280.

A seguir, a linha 290 chama as sub-rotinas que dão as cartas e a linha 300 desenha a segunda carta da banca.

A linha 320 verifica se o jogador tem um "natural" (isto é, 21 pontos em duas cartas), depois da pausa criada pela linha 310. Se houver um "natural", a linha 650 informará ao jogador, e o jogo começará de novo.

A linha 330 cuida da posição da carta seguinte. Se o jogador tiver um ás, o maior dos seus totais será **PT**. Se não houver ases, o menor (ou único) total será **PL**. Se este for maior que 21, a linha 680 informará ao jogador que houve um "estouro", e o programa irá para a linha 750. A linha 350 verifica se o jogador tem uma "mão de cinco". Neste caso, o indicador **PS** será ativado e uma pausa será provocada antes de o jogo prosseguir.

A linha 370 mostra o total dos pontos do jogador — dois valores, se houver um ás e o valor menor for inferior a 12. A aposta é mostrada pela linha 380 e as fichas remanescentes são indicadas pela linha 390.

AS OPÇÕES DO JOGADOR

As opções de "parar", "comprar" ou "pedir" cartas são oferecidas ao jogador pelas linhas 400 a 420. As linhas 430 a 450 cuidam das respostas. Se o jogador tentar comprar cartas sem ter fichas suficientes para dobrar a aposta, a linha 460 dará o aviso. A linha 470 atualiza o valor da aposta e do total de fichas, antes que a linha 480 dê outra carta.

Se o jogador tentar "parar" sem ter pelo menos dezesseis pontos, será impedido pela condição **PT < 16** na linha 430. A linha 490 é desnecessária e só entrará em ação caso essa condição seja modificada.

A linha 750 diz ao jogador quantas fichas sobraram e verifica se ele conseguiu quebrar a banca. Se isso acontecer, haverá um efeito sonoro depois de uma mudança de cores.

As linhas 760 a 780 permitem ao jogador uma outra "mão".

COMO COMBINAR PROGRAMAS

Uma das características da programação consiste em que tudo o que não esteja gravado em fita ou no disquete tem que ser digitado. Freqüentemente, contudo, pode-se economizar tempo, modificando programas já prontos ou combinando dois (ou mais) programas para formar um terceiro.

Existem basicamente duas maneiras de combinar programas. A primeira consiste em somar duas partes com números de linhas diferentes. A segunda, mais complexa, permite a união de programas com números de linhas iguais. Neste curso, os dois métodos serão tratados simultaneamente.

QUANDO COMBINAR

Combinar programas (*Merge* é o termo usado em inglês) é essencialmente um método de auxílio à programação. Suponhamos, por exemplo, que você tenha feito um programa que não funciona do modo esperado, ou que deseje alterá-lo para que ele execute uma tarefa diferente daquela para a qual foi criado. A melhor maneira seria gravá-lo em fita ou disquete e continuar a trabalhar nele sem medo de danificar o original. Uma vez aperfeiçoado, o programa estaria, digamos, com o dobro do tamanho original, ou teria ganho vários módulos diferentes. Em qualquer caso, as duas versões podem ter elementos que você deseje conservar, mas existem entre elas tantas diferenças que seria muito difícil digitar cada coisa separadamente. E há mais um problema: a não ser que você coloque os dois programas no papel, é necessário ter tudo na memória para poder fazer as mudanças na hora de combiná-los.

Outra situação em que se torna necessário combinar programas é aquela em que se deseja incorporar rotinas ou procedimentos já existentes a um novo programa. Estes podem ser longas rotinas em código de máquina que tocam uma música, ou uma rotina que anima um desenho em alta resolução, ou mesmo uma rotina de detecção de erros.

A maioria dos programadores tem uma variada "biblioteca" de sub-rotinas, montadas ao longo dos anos e incorporadas aos novos programas de acordo com as necessidades. Uma "biblioteca" desse tipo tem um valor inestimável pois, uma vez que as rotinas tenham sido digitadas, testadas e gravadas, não é necessário nem mesmo desejável redigitá-las toda vez que for preciso utilizá-las. Elas podem ser incorporadas ao programa por meio das técnicas de combinação.

Esse recurso serve igualmente para enfileirar uma série de pequenos programas e executá-los seqüencialmente. Neste caso, pode-se ter, por exemplo, uma série que comece por uma saudação inicial, seguida de um desenho multicolorido, e assim por diante.

A última linha pode desviar o programa para o início, provocando uma execução contínua.

Por que perder tempo digitando rotinas e programas já digitados e testados, se o computador pode fazer todo esse trabalho com a ajuda apenas de alguns comandos?

- POR QUE JUNTAR PROGRAMAS
- COMO ACRESCENTAR SUB-ROTINAS PRÉ-FABRICADAS
- COMO COMBINAR VÁRIOS PROGRAMAS EM UM SÓ

COMO COMBINAR

Cada computador tem seu próprio método para fazer a combinação de programas. Alguns dispõem de um comando (**MERGE**) que se encarrega da tarefa, fazendo com que o programa que está na memória seja retido enquanto outro é carregado da fita ou disquete. Já outros exigem que se digite algumas linhas de instrução, ou se destine espaço na memória para o segundo programa.

Qualquer que seja o método utilizado, no entanto, é necessário que os números de linhas sejam ajustados antes para que a combinação se dê da forma desejada.

O ZX-81 é um caso especial. Ele não tem um comando para fazer a combinação e não há nenhuma técnica razoavelmente simples que permita que dois programas sejam carregados na memória. Veja a seção *Perguntas e respostas* para maiores informações.

S

No Spectrum, a combinação de programas é feita por intermédio do comando **MERGE**, que torna muito simples essa operação. No entanto, é necessário atenção com a numeração dos programas a serem combinados. Por exemplo, para juntar uma sub-rotina de umas vinte linhas a um programa maior, é preciso deixar um espaço na numeração. Assim, o programa seria numerado de 10 a 50, digamos, e de 300 a 1000, e a sub-rotina de 60 a 250.

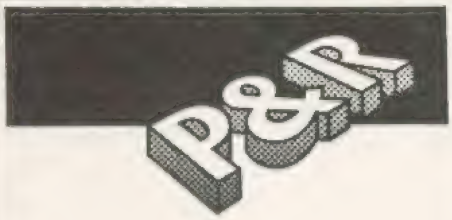
Outra possibilidade é dar à sub-rotina uma numeração superior à do programa. Com um pouco de cuidado, pode-se arranjar as coisas de tal forma que, com a substituição de algumas linhas por outras, e a incorporação de novas linhas, obtenha-se um novo programa.

Agora, digite e execute o programa a seguir. Ele mostra quadrados coloridos em posições aleatórias. Mais adiante, você verá como combiná-lo com outro programa:

```
 10 BORDER 0: INK 9
 50 PAPER 0
 60 CLS
 70 FOR n=1 TO 400
 80 LET x=INT (RND*32)
 90 LET y=INT (RND*22)
100 PAPER 7: IF x>8 AND x<24
AND y>6 AND y<16 THEN  PAPER 4
110 PRINT AT y,x;" "
120 NEXT n
```

Grave o programa (com o título de SQ1, por exemplo). E não desconecte o gravador — você ainda vai precisar dele. Note que qualquer outro programa que você já tenha serviria para nossa demonstração. Digite **NEW** e em seguida o próximo programa:

```
 10 BORDER 0: PAPER 0: INK 9:
CLS
 30 FOR t=1 TO 5
 40 INK INT (RND*6)+2
 60 PRINT AT t,0;"ISTO E UM TE
 STE"
130 NEXT t
```



Existe uma forma simples de combinar programas no ZX-81?

Infelizmente, não. Isso pode ser feito em linguagem de máquina, mas o programa é bem complicado.

O ZX-81 não tem um comando **MERGE** e não existe nenhuma forma em BASIC de carregar dois programas na memória ao mesmo tempo (todos os programas são carregados no mesmo endereço da memória). Assim, ao se carregar um programa, inevitavelmente se destrói o anterior.

Este é um programa muito simples, que mostra uma mensagem (linha 60) cinco vezes. Agora digite **MERGE""** para combiná-lo com o anterior (não se esqueça de retroceder a fita até o início do programa). O programa da fita será então adicionado ao da memória do micro. Liste o novo programa e observe que as linhas 10 e 60 foram substituídas, enquanto as linhas 50 e de 70 a 120 foram acrescentadas. Execute o programa para ver o procedimento inicial (SQ1) executado cinco vezes.

Como não é muito fácil renumerar programas no Spectrum, convém ajustar a numeração das linhas antes de combinar os programas.

T

O TRS-Color permite a junção de programas em seqüência, mas não a sobreposição de linhas.

Digite e execute o programa seguinte, que coloca quadradinhos coloridos em áreas retangulares da tela.

```
10 PCLEAR 4
20 FOR T=1 TO 5
30 CLS 0
40 FOR N=1 TO 400
50 X=RND(32)-1
60 Y=RND(16)-1
70 IF X<6 OR X>25 OR Y<4 OR Y>1
1 THEN C=175 ELSE C=255
80 POKE 1024+Y*32+X,C
90 NEXT
100 NEXT
```

Agora, grave o programa (como SQ1, por exemplo), para que ele possa ser carregado novamente na memória. Ele poderia ser uma longa sub-rotina ou parte de um programa inacabado.

Digite o comando **NEW** e, em seguida, o próximo programa:

```
10 PCLEAR 4
20 CLS
30 PRINT @192,STRING$(32,CHR$(2
36));
40 PRINT " BENVINDO A UM DISPLA
Y COLORIDO"
50 PRINT STRING$(32,CHR$(163))
```

O programa imprime uma mensagem na tela (linha 40). Para juntar a ele o programa que você já gravou, entre as linhas a seguir:

```
POKE 25,PEEK(27)
POKE 26,PEEK(28)-2
```

Em seguida, carregue o programa da fita (SQ1) e entre estas instruções:

```
POKE 25,30
POKE 26,1
```

Ao listar o novo programa, ele aparecerá com números de linha de 10 a 50 e a seguir de 10 a 100. Digite **RENUM** para renumerar o programa e liste-o novamente; você verá algo como:

```
10 PCLEAR 4
20 CLS
30 PRINT @192,STRING$(32,CHR$(2
36));
40 PRINT " BENVINDO A UM DISPLA
Y COLORIDO"
50 PRINT STRING$(32,CHR$(163))
60 PCLEAR 4
70 FOR T=1 TO 5
80 CLS 0
90 FOR N=1 TO 400
100 X=RND(32)-1
110 Y=RND(16)-1
120 IF X<6 OR X>25 OR Y<4 OR Y>
11 THEN C=175 ELSE C=255
130 POKE 1024+Y*32+X,C
140 NEXT
150 NEXT
```

A primeira parte do programa é executada rapidamente. Para separá-la da segunda parte, mude a linha 60 para:

```
60 FOR K=1 TO 2000:NEXT
```

MSX

O MSX é outro computador que apresenta grande facilidade para combinação de programas. Como o Spectrum, ele possui o comando **MERGE**, que se encarrega de todo o trabalho.

Todavia, é sempre importante tomar cuidado com a numeração das linhas, pois, se existirem linhas de mesmo número nos dois programas a serem unidos, aquelas que vêm do programa que é carregado da fita substituirão as da memória. Assim, deve-se sempre deixar espaço suficiente no programa que vai receber o novo trecho para que não se percam linhas.

Digite o programa apresentado a seguir. Ele desenhará quadradinhos coloridos em determinada região da tela. Em seguida grave-o na fita, usando o comando **SAVE** (e não **CSAVE**). Chame-o, por exemplo, de SQ1.

```
60 R=RND(-TIME)
70 SCREEN3
80 COLOR 15,R*15
90 FORI=1TO400
100 X=INT(RND(1)*256)
110 Y=INT(RND(1)*192)
120 IFX<360RX>210ORY<200RY>110T
HEN100
130 PSET(X,Y),RND(1)*15
140 NEXT
```

Quando tiver terminado a gravação do programa, digite **NEW** e, sem desconectar o gravador do micro, coloque este outro programa no seu computador:

```
10 CLS
20 PRINTSTRING$(39,36)
30 PRINTTAB(10);"VEJA QUE BELO
EFEITO!"
40 PRINT:PRINTSTRING$(39,36)
50 FORJ=1TO500:NEXT
150 SCREEN0
160 PRINTSTRING$(39,123)
170 PRINTTAB(10);"ACABOU!!!"
```

Este é um programa muito simples, que coloca duas mensagens na tela. Sozinho, não quer dizer muita coisa; mais adiante, porém, você poderá avaliar sua importância.

Agora, rebobine a fita para posicioná-la no início do programa anterior. Digite **MERGE"SOL"**. Em seguida, digite **LIST**. Veja que os dois programas estão juntos, o primeiro colocado dentro do segundo.

Como o MSX também oferece o comando **RENUM**, você não terá dificuldade para fazer com que os números das linhas dos programas sejam os mais adequados às combinações que quiser fazer.

Apple

O Apple II não tem o comando **MERGE** no BASIC; mas, para quem trabalha com uma unidade de disco, existe um programa no disco-mestre do sistema operacional que renumera as linhas de um programa em BASIC e permite que se faça a combinação de programas. Para trabalhar com ele, basta seguir as instruções do manual.

Aqui, mostraremos duas outras formas de se combinar programas. A primeira é destinada a quem trabalha com disquetes. Essa técnica aproveita a facilidade do Apple para ler um arquivo texto, simulando entradas de teclado. Pode-se guardar, assim, uma série de instruções (uma rotina de ordenação, por exemplo) na forma de um arquivo

texto e, por meio do comando **EXEC**, fazer com que a rotina seja adicionada ao programa da memória. Tudo funciona como se a rotina tivesse sido digitada no teclado. Vejamos como isso acontece. Digite o programa a seguir:

```
1 D$ =  CHR$ (4)
2  PRINT D$;"OPEN TEXT"
3  PRINT D$;"WRITE TEXT"
4  POKE 33,33: LIST 10 -
5  PRINT D$;"CLOSE": POKE 33,40
: END
100 D$ =  CHR$ (4)
110  PRINT D$;"OPEN TEXT"
120  PRINT D$;"WRITE TEXT"
130  POKE 33,33: LIST  - 5
140  PRINT D$;"CLOSE": POKE 33,
40
```

Depois de digitá-lo, execute-o a partir da linha 100 (**RUN 100**). Isto fará com que seja criado um arquivo texto no seu disco que contém as linhas 1 a 5. Ao verificar o diretório (catálogo) do disco (**CATALOG**), você verá que o arquivo tem o nome de **TEXT**. Modifique-o para **MERGE** e você se lembrará facilmente do nome (**RENAME TEXT, MERGE**). Agora digite **NEW** e a seguir o próximo programa:

```
40  GR
50  FOR I = 1 TO 500
60  COLOR=  RND (1) * 16
90  PLOT X,Y
100  NEXT
70 X =  RND (1) * 40
80 Y =  RND (1) * 40
```

O programa mostra uma grande quantidade de quadrados coloridos na tela; é bem simples, mas servirá para a nossa demonstração. Agora digite **MON C,I,O** e depois **EXEC MERGE**. Você verá as linhas 1 a 5 do programa anterior aparecerem na tela. Liste o programa e verifique se as linhas que você havia digitado ainda estão lá. Dessa forma, você já conseguiu combinar dois programas: o que desenha na tela e o outro, que pode não lhe parecer muito claro. Agora, vamos explicá-lo melhor. A primeira linha apenas define a variável **D$** como o caractere de código 4; este indica ao computador que um comando do sistema operacional está sendo usado. A seguir, um arquivo é aberto e preparado para receber dados (linhas 2 e 3). A linha 4 coloca a listagem do programa nesse arquivo, a partir da linha 10. O arquivo é então fechado e termina a rotina.

A vantagem dessa operação consiste em que uma rotina assim armazenada pode ser adicionada a qualquer programa com apenas um comando.

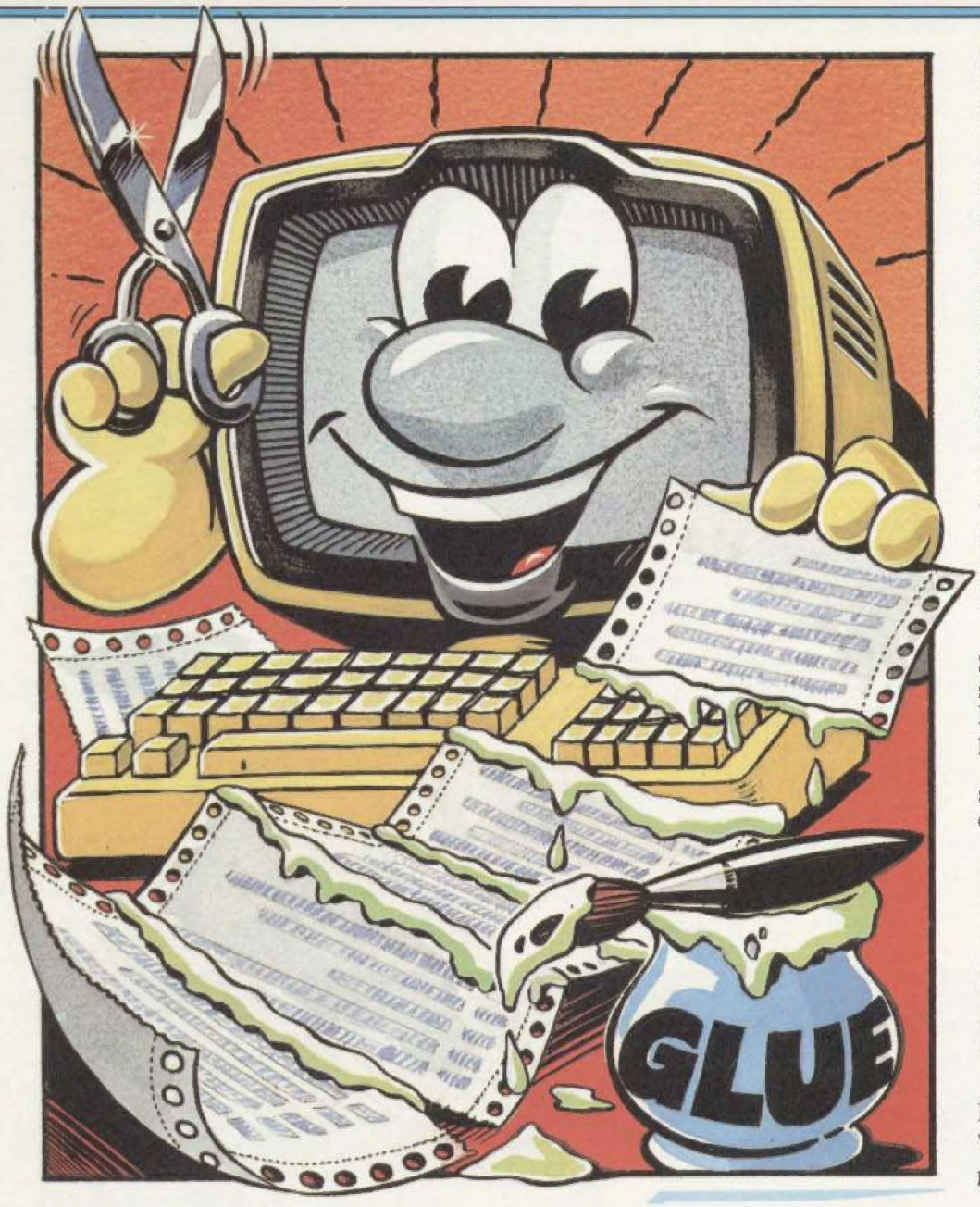
Para compreender melhor, digite **RUN**. Nossa pequena rotina será executada e o programa que coloca os quadrados coloridos na tela será armazenado no arquivo de nome TEXT. À medida que outras rotinas forem sendo guardadas, é necessário dar a elas novos nomes como foi feito com **MERGE**. Agora digite **NEW** e a seguir o próximo programa:

```
10  HOME
20  PRINT "***    DEMONSTRACAO
DO MERGE    ***"
30  FOR I = 1 TO 2000: NEXT
110  TEXT : GOTO 10
```

Este programa imprime apenas duas mensagens na tela. Para uni-lo ao programa dos quadradinhos, você deve digitar **EXEC TEXT**. Feito isso, os dois programas podem ser executados como se fossem um só.

MICRO DICAS

Se o seu micro tem um comando para a renumeração de linhas, use-o para que o programa comece em uma determinada linha e tenha um incremento constante. Se você não tem o programa, a renumeração deve ser feita manualmente; mas tome cuidado com todas as linhas referenciadas em instruções **GOTO** ou **GOSUB** e **ON...GOTO** ou **ON...GOSUB**.



SEGUNDO MÉTODO

Para quem não tem uma unidade de disco, há um outro método, mais rudimentar, que também permite a combinação de programas. Esse método coloca um programa após o outro, bloqueando a mistura de linhas de números variados. Assim, o programa da memória deve ter numeração inferior ao daquele que vai ser carregado da fita.

Digite inicialmente as linhas de 40 a 100 do programa anterior. A seguir, grave-o em fita. Limpe a memória do micro (**NEW**) e digite as linhas de 10 a 30. Não digite a linha 110. Agora execute as seguintes instruções:

```
E=PEEK(106)*256+PEEK(105)-2
POKE 104,INT(E/256)
POKE 103,E-PEEK(104)*256
```

```
Carregue o programa da fita com
 o comando LOAD

POKE 104,8
POKE 103,1
LIST
```

E temos os dois programas juntos.

Os microcomputadores compatíveis com a linha TRS-80 que dispõem de BASIC para disco (ou seja, sob o sistema operacional DOS) já têm os comandos **MERGE** — destinado a combinar programas (com possibilidade de substituição de linhas e de inserção de linhas intermediárias) — e **RENUM**, que permite a renumeração das linhas do programa resultante. Assim, é muito fácil fundir dois ou mais programas em rotinas em BASIC: basta carregar ou digitar o primeiro programa na memória (através de um comando **LOAD**) e, em seguida, digitar um comando **MERGE "nome"**, para especificar a denominação do programa a ser fundido com o primeiro.

O TRS-80 na versão cassete não tem o comando **MERGE**; para combinar programas nesse micro são necessários, portanto, alguns "truques" especiais. Esses "truques", contudo, permitem apenas que se juntem programas em seqüência, isto é, um depois do outro, impedindo a superposição de linhas.

Digite e execute o programa seguinte, que coloca uma mensagem várias vezes na tela:

```
100 CLS
110 FOR I=1 TO 5
120 PRINT "GOSTOU DO TESTE ?"
130 NEXT I
```

Agora, grave o programa (como SQ1, por exemplo), para que ele possa ser carregado novamente na memória. Ele poderia ser uma longa sub-rotina ou parte de um programa inacabado.

Digite **NEW** e depois o programa a seguir, que serve para preencher a tela com quadradinhos ao acaso:

```
10 RANDOM
20 FOR T=1 TO 5
30 CLS
40 FOR N=1 TO 400
50 X=RND(64)-1
60 Y=RND(16)-1
70 SET (X,Y)
80 NEXT N
90 NEXT T
```

Ele imprime uma mensagem na tela. Para juntar o programa que você já gravou, entre a linha a seguir:

```
PRINT PEEK(16633)
```

Se o resultado mostrado na tela for menor do que 2, digite as linhas abaixo:

```
POKE 16548,PEEK(16633)+254
POKE 16549,PEEK(16634)-1
```

Entretanto, se o resultado for maior ou igual a 2, digite estas linhas:

```
POKE 16548,PEEK(16633)-2
POKE 16548,PEEK(16634)
```

A seguir, carregue o programa da fita (SQ1), usando o comando **CLOAD**, e entre estas instruções:

```
POKE 16548,233:POKE 16549,66
```

Ao listar o novo programa, ele aparecerá com números de linha de 10 a 70 e a seguir com os números de 100 em diante. Liste-o novamente, e você verá o programa inteiro.

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81

TRS-80

TK-2000

MSX

Spectrum

TRS-Color

Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO BASIC

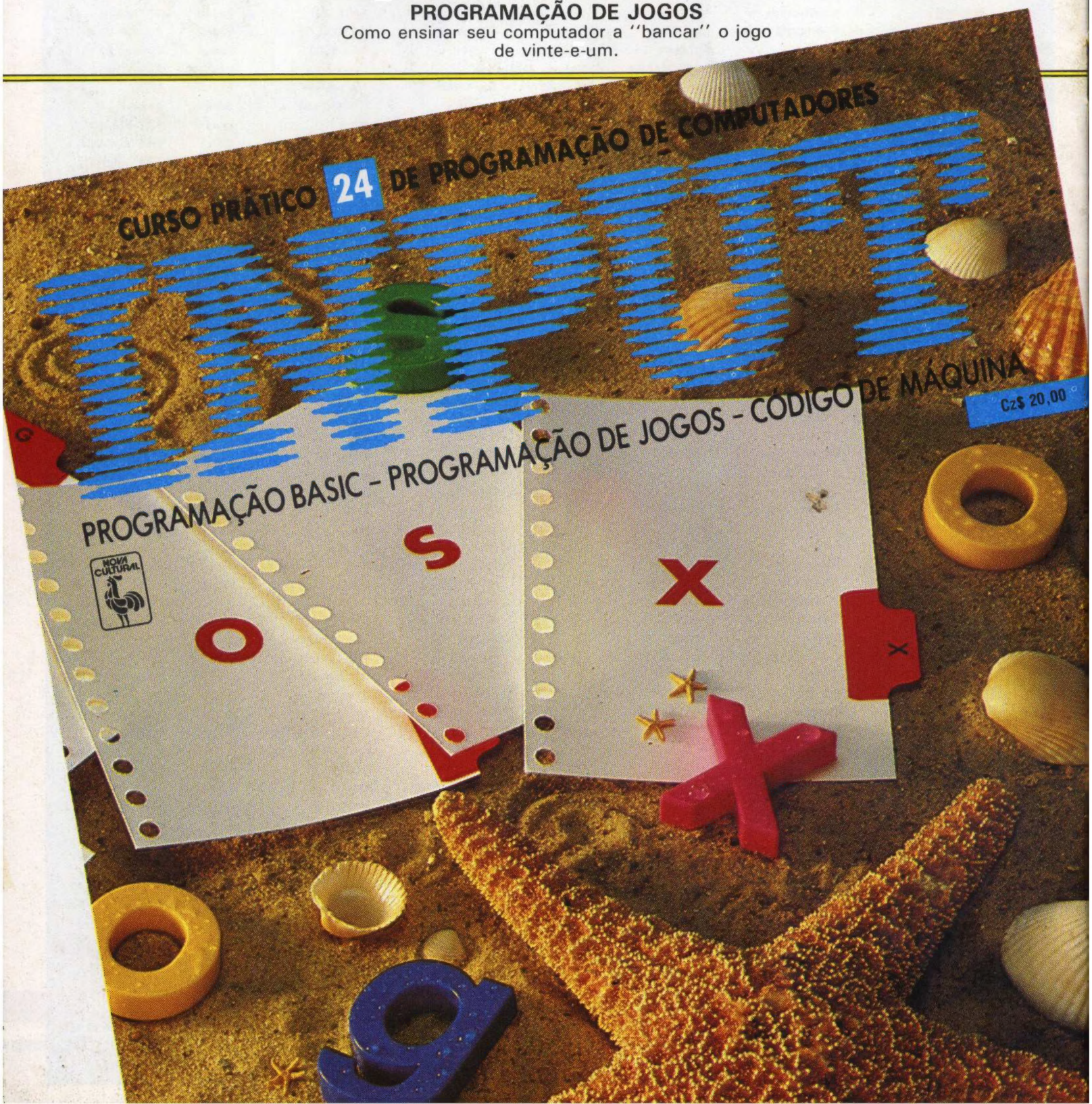
As rotinas de ordenação beneficiam todo tipo de programa que manipula dados. Três métodos infalíveis.

CÓDIGO DE MÁQUINA

Aprenda a desenhar no vídeo um dragão que cospe fogo e viva novas aventuras.

PROGRAMAÇÃO BASIC

Comandos **GET** e **PUT**: com eles você pode armazenar figuras em matrizes e trazê-las de volta à tela.

PROGRAMAÇÃO DE JOGOS

Como ensinar seu computador a ''bancar'' o jogo de vinte-e-um.